Carnegie Mellon
Software Engineering Institute

Discovery Colloquium: Quality Software Development @ Internet Speed

Linda Levine
Software Engineering Institute

Richard Baskerville
Georgia State University

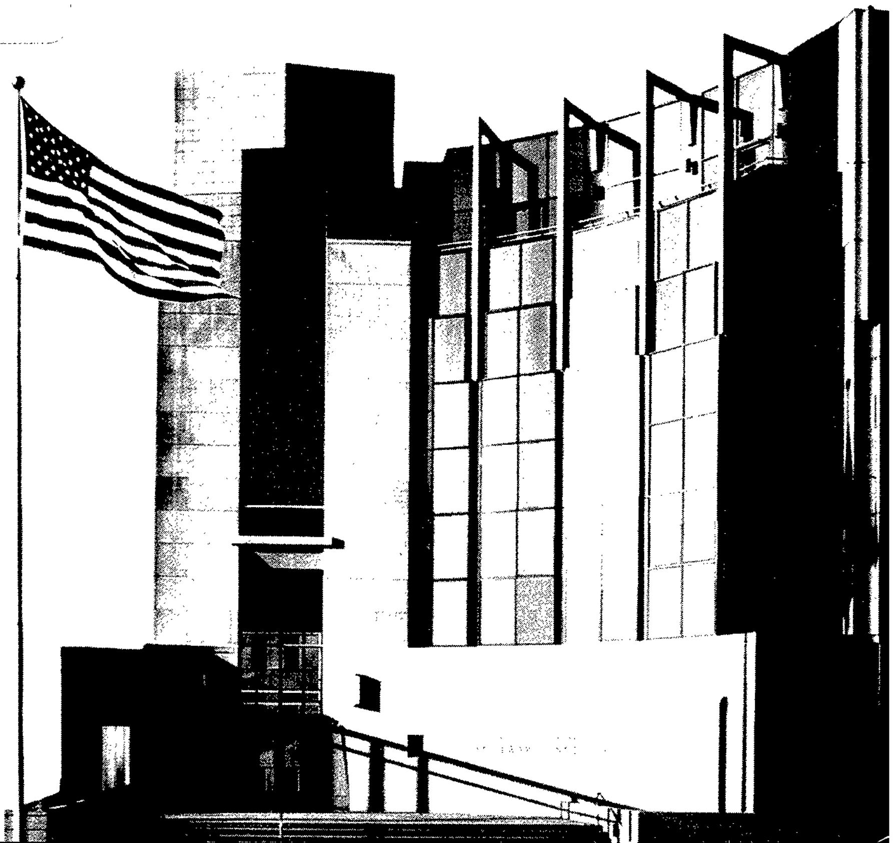
Jo Lee Loveland Link
Volvox, Inc.

Jan Pries-Heje
The IT University of Copenhagen

Balasubramaniam Ramesh
Georgia State University

Sandra Slaughter
Carnegie Mellon University

September 2002



TECHNICAL REPORT
CMU/SEI-2002-TR-020
ESC-TR-2002-020

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Discovery Colloquium: Quality Software Development @ Internet Speed

CMU/SEI-2002-TR-020
ESC-TR-2002-020

Linda Levine
Software Engineering Institute

Richard Baskerville
Georgia State University

Jo Lee Loveland Link
Volvox, Inc.

Jan Pries-Heje
The IT University of Copenhagen

Balasubramaniam Ramesh
Georgia State University

Sandra Slaughter
Carnegie Mellon University

September 2002

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Table of Contents

List of Figures

List of Tables

Acknowledgements

We appreciate John Link's and Jo Lee Loveland Link's valuable contributions to the colloquium design and their expert touch in facilitation of the event. In addition, our thanks go to Pamela Curtis and Gretchen Underwood for technical editing in the preparation of this report.

Finally, this report would not be possible without the insights and contributions of those who participated in the exploratory field study and/or the colloquium event. Necessarily, these individuals remain anonymous, although they are due our deepest thanks.

Abstract

In October, 2001, the Software Engineering Institute hosted a colloquium to explore issues faced by organizations challenged with developing quality software at "Internet speed" and to examine the impact of Internet speed on current software-development practices. The colloquium was an element of an exploratory study being conducted by a team of researchers from Carnegie Mellon University, Georgia State University, and The IT University of Copenhagen. Team members shared their findings from a previous phase of the study in which they had investigated quality and fast-cycle Internet-software engineering in nine organizations. Members of those organizations attended the colloquium, along with others who were invited to participate based on their perceived ability to contribute to the discussion of Internet-speed development.

This report describes the activities of the colloquium and the raw data collected during group discussions and breakout sessions. It also includes a brief description of the field study done by the cross-institutional team. A preliminary analysis of the results of the colloquium is presented in the conclusion of the report.

1 Background: Creating the Colloquium

The fast-moving nature of the Internet marketplace has left a gap in formal knowledge about how Internet software product development is carried out in practice. How do software developers in this exploding market actually build their fast-cycle-time software? What is the impact of these software development practices on the quality of the software?

In July 2000, a cross-institutional team commenced work related to research on Internet software development. The team included the following members:

Richard Baskerville, Georgia State University
Linda Levine, Software Engineering Institute, Carnegie Mellon University
Jan Pries-Heje, The IT University of Copenhagen
Balasubramaniam Ramesh, Georgia State University
Sandra Slaughter, Carnegie Mellon University

The team has sought to (*a*) understand how and why the development of Internet software is different from traditional software development; and (*b*) discover innovative practices used to achieve both quality and agility in Internet software development. The team's major effort has been an exploratory field study on Quality Software Development @ Internet Speed.

1.1 The Field Study

The team's exploratory field study involved investigating quality and fast-cycle Internet software engineering in nine organizations. The organizations ranged in size from 20 employees to more than 300,000 employees and were in different industries within the private sector, including financial services, insurance, business and consulting services, courier services, travel, media, and utilities. Some of the organizations were new Internet software start-up companies, while others were established "brick and mortar" companies that were implementing new Internet software development units.

The team developed and tested a standard interview script and used it to conduct an initial set of semi-structured interviews in the fall of 2000 with teams of software developers and project managers in these organizations. Interviews focused on eliciting the organizations' development methods and tools, quality issues, speed issues, product issues, and development team and people-related issues, as well as organizational demographics.

The findings from this study suggest that the practices used to develop software in the Internet environment differ sharply from traditional models of software engineering. For example, in these firms, development cycles were driven by features and were brief in duration, ranging from 15 days to 3 months, instead of the multi-year efforts addressed in traditional models of software engineering. The primary driver in each of the cases was time. In order to be useful, competitive, or interesting, software products had to be rapidly brought to market.

A similar set of drivers appeared across the cases, possibly explaining why other software process attributes like performance, cost, and quality had slipped into lower priority for Internet software development. Three prominent drivers were first-to-market, release-driven products, and fluid specifications.

When time drives, quality (along with performance and cost) assumes a second-priority position. However, the team's findings suggest that this relegation had not necessarily adversely affected these Internet software development firms. Up until the dot.com bust, all of the study cases were reasonably healthy and their products were surviving.

1.2 The Discovery Colloquium

1.2.1 Goals

In order to share findings of the field study with the participants, a Discovery Colloquium was held in October 2001 on Innovative Practices for Speed and Agility in Internet Software Development. Other goals of the colloquium were to gather additional information from experts and to test out assumptions about what the team would investigate in the next phase of their research (e.g., What issues should the team follow up on? Had the team identified the right concerns?). The activities in the colloquium were designed to expand understanding of the interface between business issues and Internet software development through open dialogue, discover and explore promising practices for Internet software development, and engage participants in a vision-based approach to identify emerging and promising models, strategies, directions, and creative approaches to address current and downstream challenges in Internet software development.

The one-day event was designed to capitalize on the rich mix of participants and allow for maximum exchange of ideas, instead of the traditional conference approach of "experts" presenting panel discussions and leading workshops for "learners." The event leveraged forward-looking methodologies, including search approaches to system learning, difference questioning, and creative abrasion. Grounded in Kurt Lewin's action research model, search conferences seek to "bring the whole system in the room" [Weisbord 95, p. 4] to exchange views and learn from one another. The search conference design utilizes difference questioning, in the belief that "when members of a group question differences...they, themselves, generate the new information" [Goldstein 94, p. 100]. This is done through "creative abrasion," a process that creates a clash of ideas while maintaining an atmosphere of interpersonal cooperation, resulting in original and innovative ideas [Leonard 99].

1.2.2 Participants

The colloquium brought together software practitioners, researchers, and leading software development experts to explore and synthesize the issues and challenges faced by organizations that must develop quality Internet software with speed and agility. The colloquium was designed to contribute to this investigation through the participation of individuals involved in the interviews at the nine companies that participated in the exploratory study, as well as others who were invited to participate based upon their companies' reputation for use of innovative practices. Participants included software practitioners from entrepreneurial small firms, large brick and mortar companies, Internet business strategists, leading software development experts, and researchers and writers on the Internet. Software development methodological perspectives also ranged from adherents of agile methodologies to adherents of more traditional software process disciplines. Participants thus represented a broad spectrum of stakeholders engaged in the critical challenges of developing quality software at Internet speed.

1.2.3 Approach

The colloquium opened with an inclusion exercise focusing on the use of metaphors to illustrate participants' images about the current Internet software development environment. This exercise was followed with a series of "fishbowl" active-listening exercises conducted in context-based groups, wherein participants discussed their own experiences. The segment concluded with a full group discussion of key issues and identification of core issues for further analysis.

In the next session, participants joined one of several breakout groups dedicated to exploration of a core issue. In the first part of this exploration, groups identified observations relating to their core issue, and then developed hypotheses about the possible associated factors and dynamics. The groups worked through a process of hypothesis testing, identifying linkages, contradictions, and inter-dependencies among the hypotheses. The groups then delved into underlying assumptions to further build dialogue from generative ideas and to allow for both divergence and convergence of insights. Principles, promising practices, and other dynamics of relevance were identified. The exercise concluded with groups reporting to the full gathering the key elements of their analysis in the breakout groups. Finally, turning to the future, all participants together engaged in "proto-scenario development" through identification of emerging conditions and impacts. Several scenarios were selected for analysis of factors enabling or constraining their emergence. This exercise generated knowledge by the participants on evolving dynamics in Internet software development, as a foundation for ongoing research, and created products of value to the participants in their business strategies and operations.

By engaging innovative approaches to discovery, the colloquium designers hoped to galvanize significant ideas, identify promising practices, and generate insights into future evolution of Internet software development practices. For example, one important insight from the Discovery Colloquium is that more traditional approaches to software engineering are based on the philosophy of rationalizing a "messy" world, with the notion that one development approach is optimal.

In Internet software development, the philosophy is that software engineering cannot rationalize a "messy" world; that is, one development approach is not optimal in all situations. Instead, Internet software development approaches must be adapted to the contingencies in a particular environment. This insight has motivated the research team to explicitly examine environmental contingencies motivating the choice and effectiveness of Internet software development practices. In addition, dialogue during the course of the colloquium revealed an agreement that basic software development principles do not change; rather, principles are implemented in specific practices that evolve and are matched to environmental contingencies.

2 Introduction and Opening Briefing

A facilitator opened the meeting by briefly describing the context for the colloquium and explaining how it arose from the ongoing study on "Quality Software Development @ Internet Speed." The specific focus of the colloquium was to provide and extend insights offered by this study, while considering potential new directions for future phases of the project.

The study team hoped to produce findings that would be useful to colloquium participants in real time. Therefore, the methodology for the colloquium was intentionally designed to parallel the emergent nature of Internet software development, and to allow for a range of different points of view to arise and be examined.

Copies of the invitation, brochure, and the agenda are included as Appendix A, B, and C, respectively.

2.1 Principles for the Colloquium

The colloquium was designed to create an environment that would foster mutual learning and sharing. The following principles were adhered to throughout:

- **All are experts / all are learners / all are contributors:** Rather than the traditional "panel discussion and Q&A," the colloquium was grounded in the belief that all participants have valuable knowledge to bring to the table. The colloquium was structured to encourage open knowledge transfer through sharing.
- **Discovery atmosphere:** To encourage sharing, the colloquium created a forum for open exchange of ideas leading to difference questioning as opposed to premature consensus.
- **Multiple factors / multiple solutions:** The colloquium was based on an understanding that complex systems are not fruitfully examined by seeking a simple root cause analysis or a single solution; rather, an inquiry into complex dynamics is necessary.
- **"Creative abrasion":** Divergent thinking can lead to deeper understanding through "sparks" generated by candid sharing of different points of view and difference questioning.
- **The field of quality Internet software development is a moving target:** The colloquium design recognized that the "History of the Internet" has already evolved through several recognizable generations, and recent events will give rise to additional changes in the future. The colloquium was set in real time, with the final exercise looking toward forecasting future scenarios.

2.2 Ground Rules

Since the colloquium was a one-day session, economical use of time was essential. The ground rules were critical to establishing a collegial climate rapidly and effectively. All participants agreed to adhere to the ground rules for the day.

- Listen to each other (allow for difference questioning)
- Take a chance—speak your mind
- Respectful disagreement is encouraged
- No soapboxes or harangues
- Suspend disbelief about the process—high levels of interaction require a tight ship
- No cell phones or laptops (except for designated scribes)—notes will be provided, and the focus is on interaction among participants
- Think out of the box
- Enjoy and have fun

3 Inclusion Exercise

3.1 Metaphors for Internet Speed

This initial exercise was designed as an inclusion exercise to begin creating the colloquium community, unearth underlying images and issues related to Internet challenges and rewards, and capture the energy and enthusiasm that participants brought to the event.

In small groups, participants identified metaphors for Internet-speed development. The following were identified:

- basketball game—fluid, dynamic, agile players, situational awareness
- zeppelins and jets—Before, slow and predictable. Now fast and unpredictable.
- building a house in 4 hours—built from scratch, scrambling for discipline, looking for external tools to create the discipline
- racing—racer adopts engine of previous winner, performance only improved when the engine is tuned to fit strengths
- changing tires when you are going 65 mph on the Interstate
- Pinocchio—team keeps telling clients "we'll have it ready for you tomorrow" while Pinocchio's nose keeps growing
- out on a limb with the customers sawing off the limb
- building a house—living in a house as it is being built
- an organism that is growing and adapting
- war
- handcrafting a Bentley
- living under a fire alarm
- mountain climbing or special forces moving towards a goal
- playing a piece of music—there's a lot of difference between a large symphony orchestra and a smaller jazz band; if musicians don't know the ground rules it will become a cacophony
- Lucy in the chocolate factory
- a ship at sea—The good ship nearly being hit by lightning; loosing control to more powerful ocean; tennis, carnival, movies going on aboard the ship; surrounded by an octopus (multiple customer demands) and sharks (competitors). Figure 1 shows a drawing created by the group to depict these issues.

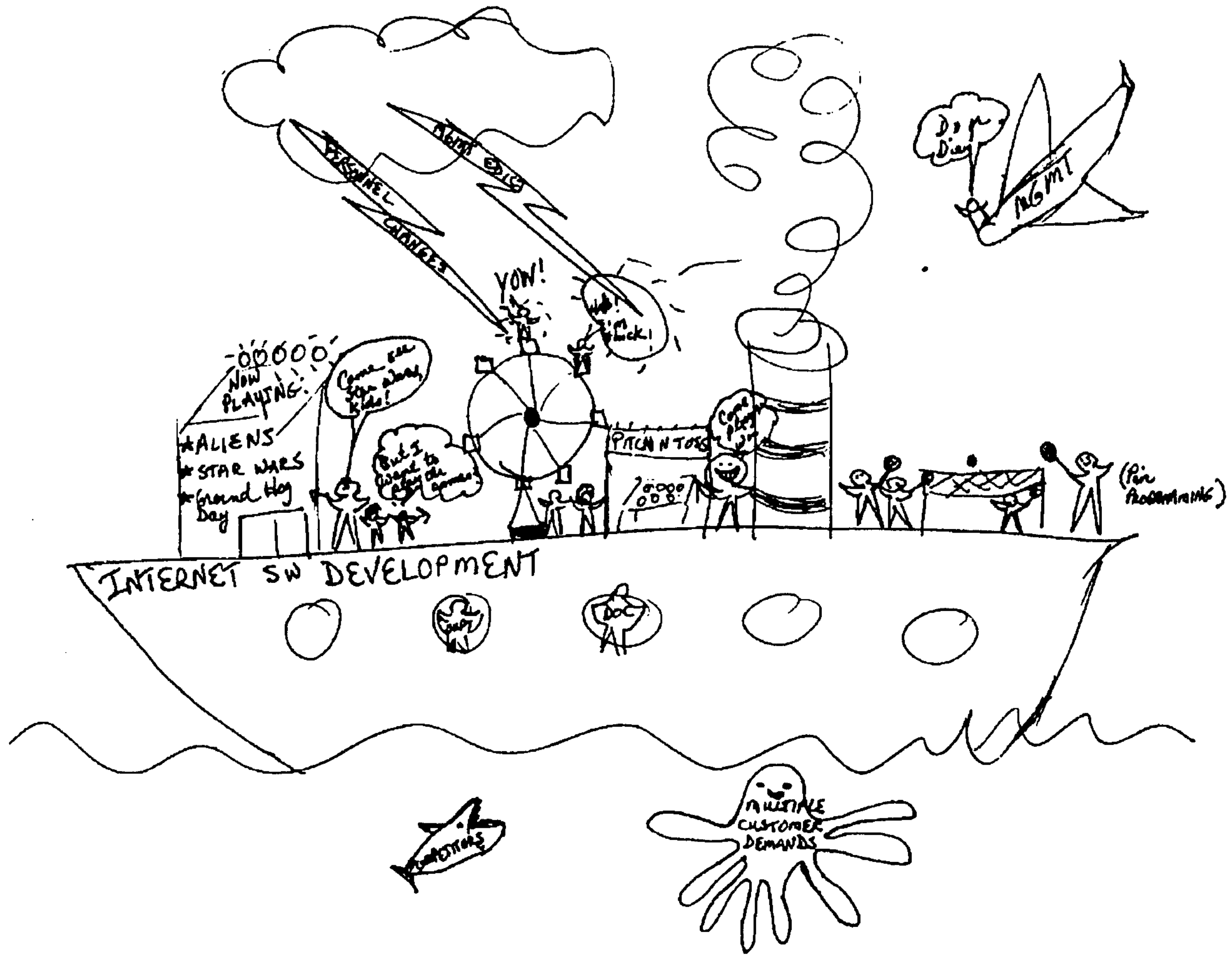


Figure 1: An Internet Speed Development Metaphor

3.2 Conclusions

The purpose of this exercise, as described above, was to establish a participant community. Creating a viable, lively colloquium community was vital to the high level of interaction and openness expected of the participants throughout the day. In this exercise, participants learned a lot about one another's thought processes and perceptions. Common themes that appear to have emerged from this initial exercise include

- danger
- movement
- change
- the need to understand prior context/environment

4 Zuni Fishbowls

4.1 Fishbowls: Listening Exercise in Sequenced Groups

The Discovery Colloquium employed a type of group reflective dialogue that integrates approaches from social science and Native American tradition. The exercise used "fishbowl" discussions as applied in education and human development, in combination with a model drawn from Zuni tribal council "talking circles."[1] A series of small participant groups (or subgroups) were centered in the larger group, discussing issues pertinent to quality software development. While each subgroup engaged in dialogue, participants in the rest of the room reflected and listened to the discussion without comment.

This purpose of the exercise was to allow a series of groups—with different contexts and backgrounds—to discuss freely, and without debate, their perceptions and knowledge of Internet software development. The two primary goals of the exercise were (1) to create an environment in which thoughtful listening was at least as important to all participants as talking and sharing, and (2) to begin framing core questions for consideration by the colloquium.

Throughout this exercise, the principle of honoring different points of view and openness to others was reinforced. Participants were encouraged to listen to each fishbowl in turn (and members of fishbowls to listen to one another). This exercise provided insight into the myriad issues relevant to Internet software development.

The fishbowls were set up in series, as follows:

1. the Quality Software Development @ Internet Speed study team
2. smaller Internet companies
3. larger Internet companies
4. a hybrid group of people who had experience in both dot.coms and brick and mortar companies with Internet components (including some consultants)

In order to provide readers a sense of the participants' unique backgrounds, the following information is provided about each of the participating organizations in Table 1: a brief description of the organization, its date of origin, and its approximate size. (Note: the latter re-

1 Loveland, J. & Loveland Link, J.L. Unpublished adaptation of a model from Zuni tribal council talking circles, 2001.

flects the size of the entire organization and is not restricted to the number of employees in the IT department alone.) The names of the participating organizations are not revealed here or elsewhere in this report, so as to enable the candid disclosure of the participants' comments.

Table 1 Participant Characteristics Chart

Company	**Brief Description**	**Date of Origin**	**Approx. Size**
A	Develops and markets a platform of e-business software modules. Fully integrated modules allow users more control by providing tools to conduct business online.	Late 1990s	8
B	Offers customers a complete solution for managing their software development life cycle. Their main product is designed to enable users to take control of their entire software development and testing process in a single, integrated environment.	Mid 1990s	5
C	Full service Internet consulting firm, helping businesses and organizations to grow and develop their potential using new technologies. Clients range from accounting firms to multimedia artists.	Late 1990s	5
D	Builds and manages technologies for clients in the financial services industry. Work has expanded beyond traditional consulting services into strategy, creative and technological design, integration, implementation, managed hosting, and the development of various software solutions.	Late 1990s	100+
E	Works with hardware/software, financial, wireless, and healthcare providers. Offers tools to enable collaboration without regard to geographic boundaries.	Late 1990s	100+
F	Provides customized development of automated rich-media applications and scalable solutions that allow media and entertainment companies, as well as enterprises and government organizations, to deploy, manage, and distribute video content on Internet Protocol (IP)-based networks.	Early 1990s	200+
G	Provides end-to-end technology consulting services to banking, financial services, insurance, telecom, manufacturing, retail, transportation, healthcare, utilities, and government institutions.	Early 1980s	16,000+
H	Offers a wide range of products and services (including training) to keep nuclear power plants operating safely and competitively worldwide.	Late 1800s	N/A
I	Targets sectors of the enterprise and service provider markets. Provides products that enable computing devices to access computer networks. Provides IP-based access and infrastructure and service platforms for the telecommunications service provider market.	Late 1970s	8,000
J	Global consulting for consumer/industrial products, financial, and technology/entertainment focused organizations.	Late 1800s	150,000
K	Multi-disciplined, global asset management firm offering diverse investment vehicles.	Late 1950s	3,000+
L	Identifies online potential for brick and mortar companies and helps them launch a dot.com site. Also helps clients to brand their products.	Late 1990s	55

4.2 The Quality Software Development @ Internet Speed Study Team Fishbowl

The study team began their fishbowl with a prepared presentation of the goals, methods, and findings from their study involving nine U.S. companies. The team described the interviews they had conducted with senior managers, project managers, software engineers, and QA engineers at those companies, examining

- organization and interviewee demographic information
- Internet speed
- product and business strategy
- quality
- teaming and people
- development methods and tools
- issues, problems, and challenges

The team reported on several key characteristics of software development at Internet speed that they had uncovered, which are depicted in Figure 2.

The team also clarified that the study was ongoing: team members were interested in emerging trends and evolutions in Internet speed and Internet software development (for a complete copy of the presentation, see Appendix D).

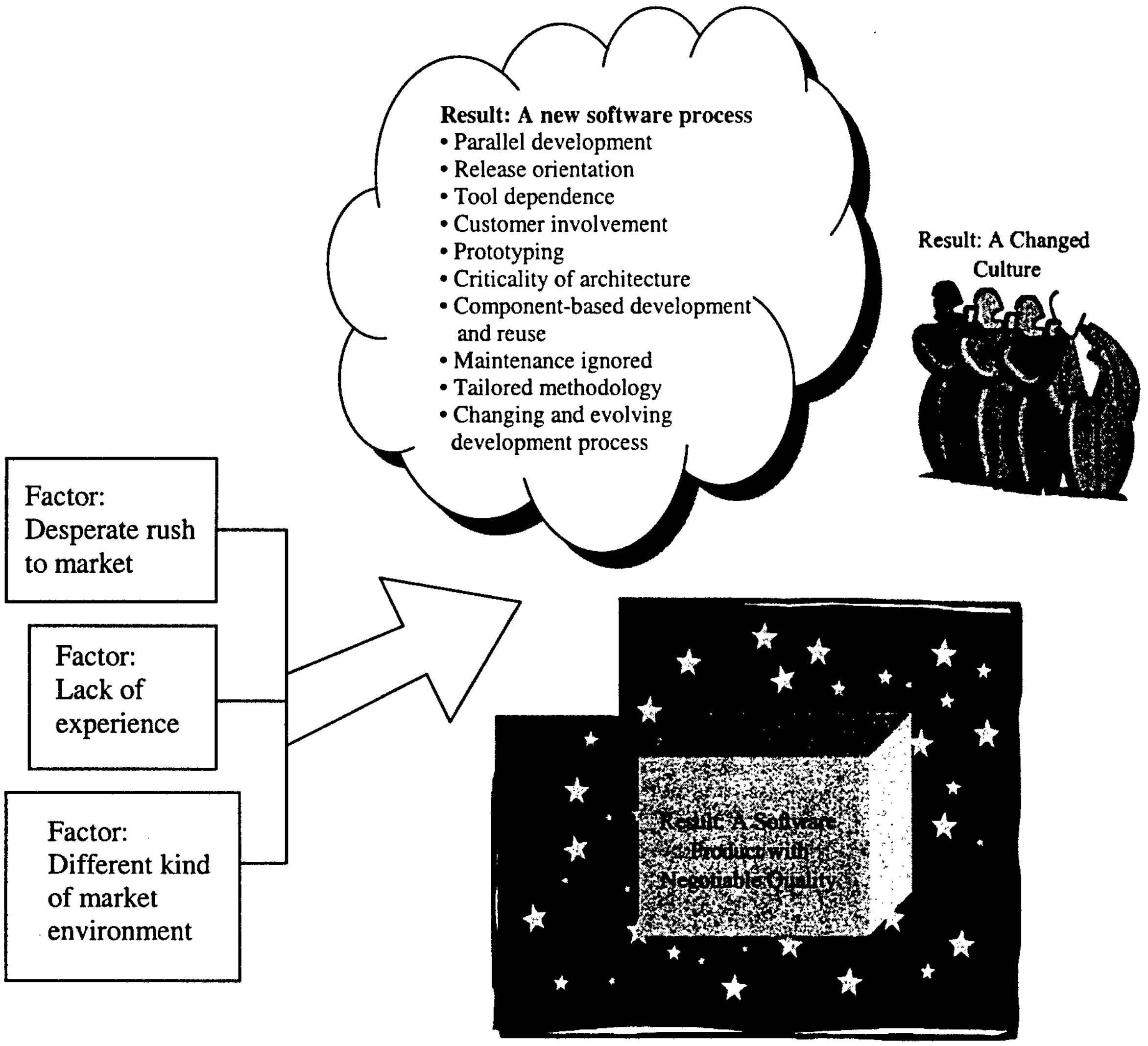

Result: A new software process
• Parallel development
• Release orientation
• Tool dependence
• Customer involvement
• Prototyping
• Criticality of architecture
• Component-based development and reuse
• Maintenance ignored
• Tailored methodology
• Changing and evolving development process
Result: A Changed Culture
Factor: Desperate rush to market
Factor: Lack of experience
Factor: Different kind of market environment
Result: A Software Product with Negotiable Quality

Figure 2: Six Characteristics of Software Development at Internet Speed

4.2.1 General Questions

Following the prepared presentation, the study team responded to questions from the facilitator and the other participants.

Q What surprised you?

A1 Negotiable quality.

A2 Quality can be product based, process based, need based—this seems to be the core.

A3 It was quite neat to see IT come up on top of the table, driving corporate strategy, rather than stuck in the backroom; taking it to the extreme when developers are developing strategies.

A4 There is no one method that works on different versions of product. A team has to continually reengineer, within a matter of weeks.

A5 When we first came together to raise the question of whether Internet speed development is different, we were at a loss for where to start. One possibility was to generate factors where it may be different in Internet development. We generated a series of hypotheses.

A6 We heard an Internet year is one to two calendar months. We couldn't believe it was so different, but the speed is really influencing everything.

A7 Speed is changing from project to project. Fast is not necessarily always good—there are issues with a customer's ability to absorb the technology.

A8 I remember a company that told us they had a customer that asked them to slow down—the customer could not change their clients' behavior so quickly—they asked the company not to give them releases so fast.

A9 In early discussion, I thought architecture would be key, but that wasn't the case. Architecture was in there, but it wasn't the core to plans of success.

A10 The issues we heard about scale emphasized being able to respond to an instantaneous end rush.

A11 People are important, especially when there is a lack of structure. So development is really dependent on individuals, and culture is more individualistic.

A12 People and scale issues interact. We can make commitments based on the capabilities of people, and the ability to bring together the right people. Dependence on individuals constrains how quickly an organization can grow.

A13 Scalability is not about the hardware side, it's about scaling the organization relative to the number of customers.

A14 I was struck by ways in which organizations achieve parallel development. Some had teams working on different releases. Others had teams doing implementation, analyses, working with customers, etc.

A15 There are unique opportunities for some individuals to be involved in all phases of a project.

A16 The tendency of the Internet companies is to have everyone at the negotiating table—including competitors! There are a number of additional tensions:

- the concept of speed is changing
- focus is not so much on product quality but need-based quality
- Internet is in "dog years"
- different solutions are used to achieve parallel development

Q Are Internet organizations working on development in their first year or two?

A Some of them have been there since early stages. Even the bricks and mortars. The two huge organizations had separated out new divisions to do their Internet development.

Q Did the study team see any significant difference among Internet organizations?

A1 The differences we saw were the significant differences between the guys in the garage vs. those who have already established some type of structure.

A2 Two of the companies were in the garage development era—they are gradually changing and developing structure.

A3 Brick and mortar companies have had to integrate their practices into their original culture.

A4 Some organizations were questioning how to make small pieces of experimental work scale and function.

Q What are we going to see ten years from now as the best Internet practices?

A It's really hard to speculate at this point—especially as there are now new emerging factors that were not predictable before.

Q What is the speed-up factor of Internet time?

A1 How much faster? I think it's only 50% faster.

A2 Possibly up to 3–4 magnitudes faster.

Q I'm curious about visibility into what the subcontractor was doing. Was it blind trust, or do they demand visibility?

A1 Companies may not know what they want—they need and want to trust the developers to help them figure out what they need.

A2 Partnership models—outcome based. Trust is based on collaboration.

Q Is an Internet-speed company only small, or something that develops software fast? Incremental life cycle has been around for a long time.

A1 Speed operates at multiple levels. Architecture takes a long time. Application layer is closer to company strategies.

A2 The business strategy is emerging with the software strategy.

A3 Low barriers to entry make speed critical.

A4 Right lines of code (LOC) /time: this issue is not a given, but rather what the right lines of code are as a factor of time. Now, we don't know what the right LOC is, and time is very short. It's a different kind of activity in that way.

Q Observation: a lot of forces affecting dot.coms are beginning to seep into the brick and mortar type companies. Do you see this, and if so, what are the implications?

A1 A lot of discussion on methodology has arisen from such different environments—eXtreme Programming, etc.—and trying to generalize these to all types of environments.

A2 Methodologies are not a waterfall. They function as agile methodologies.

A3 Flexibility in methodologies seems to be emerging—one size does not fit all.

Q In other markets, we see experimentation and learning based on trial and error in establishing companies. In Silicon Valley, we see people in their third dot.com efforts. These folks seem to have matured with their experience. Did you see that?

A We haven't followed individuals in the study to date.

Q Did companies worry about the impact of mixing development and process methods between Internet and others?

A1 Saw cases of isolation of development team, worried that they would "contaminate" the team.

A2 Question of the level of information sharing and integration.

Q Software process improvement and traditional models may not fit Internet speed companies. Is this validated by the study?

A Many companies deliberately isolate their Internet development team because they are worried that the traditional development projects would "contaminate" their Internet team.

4.3 Smaller Organizations Fishbowl

The second fishbowl included approximately seven people and was made up of small entrepreneurial Internet company members. Profiles of the participants who offered comments in this section are shown in Table 2.

Table 2 Profiles of Smaller Organizations

Company	Brief Description	Date of Origin	Approx. Size
A	Develops and markets a platform of e-business software modules. Fully integrated modules allow users more control by providing tools to conduct business online.	Late 1990s	8
B	Offers customers a complete solution for managing their software development life cycle. Their main product is designed to enable users to take control of their entire software development and testing process in a single, integrated environment.	Mid 1990s	5
C	Full service, Internet consulting firm, helping businesses and organizations to grow and develop their potential using new technologies. Clients range from accounting firms to multimedia artists.	Late 1990s	5

Company A: Our organization has eight people today. It started as three people in a spare bedroom, following the hero-commitment approach. Now that we are growing we realize that we need to be more process oriented.

Company B: Our organization was started by a person who came out of the quality movement. The idea was to run the company with a process-oriented approach from the beginning; however, there were a lot of things that were different. Essentially the people who weren't able to deal with structure and process weren't allowed in the door. So the perspective of a quality software development process was there from the beginning. I remember interviewing people that were really hot, really good, but they just wouldn't fit.

We have a culture that is different from other dot.coms. What this means is that new people have to evolve and they have to learn their own way of adapting. They must also understand how they impact the environment. Our development shop was in Ireland. Offshore development actually insulated the developers from customer pressure, because developers could not go out and see what was going on in the market. This provided us with some stability. We have managed to stay at 15 people for 5 years.

Company C: We came about from a Wild West standpoint. We had a vision. We hold quality in high regard. We thought the company was going to do well and we'd reap the benefits. Then came the torpedoes: first-to-market, get-it-out-there, etc.

Some parts of our house have done well in relation to quality. Other parts haven't. It often depends on the customers—if they are organized in their thought process, it helps us a lot. If a customer wants to build a product, but they don't know what they want, we have to slow down, and go through a discovery phase. Some customers ask for something tomorrow and we have to tell them it will suck. It's painful, some customers respond to that, but others don't want to wait. We have customers that we delivered to very fast, they call us every day, because they never knew what they wanted, and their software is much too complicated. We are slowing things down now.

4.3.1 General Questions

- What is it about quality software development at Internet speed?
- Is quality important?
- If so, what does quality mean to you?

Company A: It is true that customers don't always know what they want, they ask for more the more they see. You need to do project management; but quality is one of those words I don't use a lot, because if you ask three people you get four answers.

We are thinking of growing the business a bit more. An individual that we employed has helped us become a lot more structured. If we want to be successful in the long term, we have to act in the ways that successful companies act.

Venture capitalists are becoming more interested in the companies that they invest in and want to make sure these companies have processes in place. One venture capitalist asked the other day: Have you heard about this thing called CMM [Capability Maturity Model]?

Company C: More free flow or more structured, which will ultimately be more successful?

Company B: We have moved to two releases per product per year. We have four products per year, because our customers can't adopt any faster. We found we had to slow things down. The observation is we could have done some things differently, but we would have had to hire different people. We interviewed people who were good, but they just didn't fit. We have not grown beyond five people. We have been able to grow products without increasing staff. We're not ever going to be a 300- or 500-person company. It has to do with goals. Do you want to be local, or go out and take over the world?

4.3.2 Facilitator's Question

Q: What is your most interesting success in the past 18 months?

Company A: Trying to grow from a three-man shop into more structured company; hiring more people. We have now started developing software in a more predictable way. It is nice to see that we are getting more aligned, trying to get away from "cold pizza, warm coke." I am sleeping better, as we move away from the hero scenario.

Company C: Our biggest success was to build relationships and strategic partnerships with other organizations to give us a net to lean on. If you're on your own, you are vulnerable, like the ship alone on the sea. Now, we can approach other companies with good reputations for certain things, e.g., project managers. So if we move into markets that require such skills, it's helpful. We are finding more opportunities with these relationships. We struck a relationship with a European company that develops new versions in-house. We feel we're gaining some ground.

Company B: Our biggest success in the last 18 months was probably to move from Lotus Notes to being a Microsoft Partner, and earning awards from Microsoft.

4.4 Larger Organizations Fishbowl

The third fishbowl consisted of nine people, and was made up of Internet companies that were still entrepreneurial, but larger in size. Profiles of the participants who provided comments are shown in Table 3.

Table 3: Profiles of Larger Organizations

Company	Brief Description	Date of Origin	Approx. Size
D	Builds and manages technologies for clients in the financial services industry. Work has expanded beyond traditional consulting services into strategy, creative and technological design, integration, implementation, managed hosting, and the development of software solutions.	Late 1990s	100+
E	Works with hardware/software, financial, wireless, and healthcare providers. Offers tools to enable collaboration without regard to geographic boundaries.	Late 1990s	100+
F	Provides customized development of automated rich-media applications, and scalable solutions that allow media and entertainment companies, as well as enterprises and government organizations, to deploy, manage, and distribute video content on IP-based networks.	Early 1990s	200+
G	Provides end-to-end technology consulting services to banking, financial services, insurance, telecom, manufacturing, retail, transportation, healthcare, utilities, and government institutions.	Early 1980s	16,000+

4.4.1 General Questions

- What is it about quality software development at Internet speed?
- Is quality important?
- If so, what does quality mean to you?

Company D: Quality is highly dependent on the context of the situation. If we were to produce technology that monitors heart patients, quality would be different from the technology to sell dog food on the Internet. We service clients in the financial industry. Depending on the context, clients A and B may have different software needs: what they want may or may not be mission critical. Quality levels are dependent on the particular scenario.

We started, from day one, developing a framework: we were trying to engineer quality into our products instead of testing it in at the end. We need a very thorough understanding of the application domain. We hire people in the domains we work in, people who have an average of 15 years of experience. The education process is minimized and we have higher quality end processes. Quality is a founding principle.

We develop technology in a foreign market. Depending on whom you work with, it is unlikely you can tell them what quality means. We have to meld our expertise into what they define. We attach quality from a domain expertise focus.

Company E: We build products to support Web-based collaborative development. We're now doing things for companies like HP. Our original model was an application service provider building custom Web sites. The problem was developers were the customers: they have high expectations of quality, so they cannot afford to be sloppy. But try to make it repeatable. Quality is difficult when you build products one at a time. Our major goal is to make repeatable things—customize edges, but not the core—and do it at a level consistent with current staff and funding.

Company D: We have a customer orientation. The secret to meeting milestones at Internet speed is creative reinterpretation of requirements: adjusting, fudging, doing something that may work out better for the customer. Make trades with the customer: if we delay this thing, could we add another thing that we can do more easily. The whole idea is to make the customer happy; add little surprises, give them something they didn't ask for. Put yourself in the customer's shoes. Don't view them with contempt, but rather, try to delight them. Be creative about requirements negotiations.

Company F: We were doing a lot of education of our customers. Customers would see our products and then ask how they could use them, rather than them telling us what they wanted. We can educate them more down the path that we know we can fulfill.

We started as a technical company. We had to figure out how to make it more practical. We had to transition from academic to business—we do better in a solutions approach. We had a more consultative selling process that was customer focused, but we need to apprise them of their needs. We now go out to customers saying, "This is the pain you are feeling. And you may not know how much pain you are feeling." Quality to us is making the customer happy.

Company G: Time to market has always been a challenge. The scale has changed over a period of time. The same thing goes for satisfying customer needs: understanding customer requirements has always been a challenge. In this context, quality-related issues in the Internet domain are not totally new.

In order to address these issues, there have been a number of changes over a period of time. For example, a few years ago we mainly recruited engineers; however, in order to bridge the gap in business knowledge, we now recruit people with domain knowledge too.

The Internet just adds a new dimension to the perennial problems in software development. How do we encourage individuals to retain learning, and how do we ensure their continued learning? How should we capture design rationale and general "experience issues"?

We are a large organization with about 16,000 people, but not a monolithic entity. There are 17 centers, which in turn have many groups dealing with different types of software development. If a group needs to modify the processes to address requirements, they are always allowed. But we need the learning/experience from that group; in some cases we may also depute people from SEPG to the project to get a better understanding and retain the learning/experience for subsequent use in the organization.

Our experience shows that sharing of metrics and use of metrics to discuss problems makes it easier to communicate with customers and to build a relationship of trust. The fact that you are checking it and doing it in a manner in which you can track it makes a big difference in building customer confidence. Satisfying customers may not mean just compliance to requirements.

Process orientation should be part of the education of IT professionals (at the university level). Next comes induction into the company, where new employees are schooled in the culture and methods/processes of the company. This provides an environment where learning and sharing of experience is encouraged, where things do not go beyond control. For example, recently there has been a lot of outsourcing taking place. Systems are outsourced to subcontractors who maintain the system. A lot of thought had to be put into understanding what goes into maintenance and enhancement of systems developed by another organization. Now we have a better understanding and better processes to address the requirements of these maintenance and enhancement projects. It's a learning process the whole way through.

4.5 Dot.com + Brick and Mortar Fishbowl

The fourth fishbowl contained eight participants, and was a hybrid group. These individuals had experience in both dot.coms and brick and mortar companies with Internet components. Consultants were included in this group. Profiles of the participants that offered comments in this section are shown in Table 4.

Table 4: Profiles of Dot.com + Brick and Mortar Organizations

Company	Brief Description	Date of Origin	Approx. Size
H	Offers a wide range of products and services (including training) to keep nuclear power plants operating safely and competitively worldwide.	Late 1800s	N/A
I	Targets sectors of the enterprise and service provider markets. Provides products that enable computing devices to access computer networks. Provides IP-based access and infrastructure and services platforms for the telecommunications service provider market.	Late 1970s	8,000
J	Global consulting for consumer/industrial products, financial, and technology/entertainment focused organizations.	Late 1800s	150,000

4.5.1 Group Responses

Consultant 1: In delivering solutions in Internet time, we have to adapt quality and priority so that mission criticality of the applications is not affected. Organizations use different strategies and methodologies to provide this multilayered development paradigm. Quality is achieved, not through one strategy, but through multiple strategies, as deemed appropriate by the particular development segment. Very rarely in a large brick and mortar company is there a case where Internet development and legacy development do not intermingle.

Consultant 2: A company that I consult for is a large mutual fund company and is characterized by all of the things that the team found, except the negotiated quality. They invest in testing, not so much CMM processes. Quality was a requirement—fundamental to business structure. I do not think that brick and mortar companies can sacrifice quality.

Company H: Our group is small, 20 people, and focused in the nuclear industry. My company is very interested in finding ways to speed things up—like methodologies to do things better and more quickly. In the nuclear industry we must get things right, but we have to qualify models in terms of real life. We need to do this rapidly, and yet the things that we discussed seem to be a throwback from thirty years ago. I just got asked a question about code I wrote 34 years ago, when I wasn't concerned about maintainability. Dot.coms are only starting to get interested in the solution.

Comment: There is no such thing as change without negative impacts on large systems. Maybe you can manage change, and don't have to worry about maintenance in 5000-LOC products. But if you try that strategy in 1m-LOC products, that makes me nervous.

Consultant 1: I went from total brick and mortar to total startup. We started software development with a closed system. While at a major national bank we believed we could create software and service the way you build cars, that organizations are modular in design. Wrong! People change every time they do something.

The Internet has magnified the fact that we're open systems because of ways that information flows: it is wide open. We are still trying to design software for a closed system. We need to move towards models of living systems. Living systems learn. Quality is your ability to adapt. We need to develop open system organizations to create open systems software.

For our company, going from brick and mortar to startup, the CEO understood complexity. We got most of our ideas from CMM and PSP. We don't have to throw out CMM, but we need to do things differently. The important thing is stability and variability. Focus on learning. Influx of information brings variability; stability is the closed loop. We need to retain what we learned from software engineering, but throw out what we learned from building cars. eXtreme Programming gave us constant adaptation. Massive change will be random, no planning possible, but humans can project futures.

Company I: Regardless of the nature and size of the companies, the fundamental set of problems is the same. We should think about adequate quality instead of high quality. What is ultimately going to make your customer happy? Often, we miss the fact that the customer would rather have something now than have something good later. Requirements only make sense after you understand customer needs.

There is a huge need to break down barriers between customers (users) and developers. For example, we set quality standards for the Navy, and we achieve them by breaking all the rules. We look at requirements—so many pages—like we are learning English by reading the dictionary. We paired our programmer with the customer. This was unheard of in the 1980s, but we wanted to break down the wall. We invited Navy people to talk to our people. Once we developed a sense of trust, there was an "aha." My programmers can easily pick up the phone to talk to a customer.

We always talk about "the what" and "the how," but never "the why." That is why you cannot adapt "the what" or "the how" to your environment. The difference between traditional and new approaches—Charles S. Beard has shown that if you do eXtreme Programming [XP], you are close to achieving KPAs [Key Process Areas] for L3 [Level 3] for CMMs. I don't see anything new in XP. We had a military V-model, and we were doing pair programming and daily testing 20 years ago. We need to keep software development on a principle-centered approach. The Internet has made a huge difference in empowering people. Better organizations are self-organizing. The Internet bridges the distance. People with different expertise in different geographic areas can collaborate.

Consultant 1: Quality is in reference to its context. We have to think about co-evolution. The way things join is that they develop a protocol. This can be done through dialogue. This is how we create new things. That dialogue is the negotiated joining. Self-organization is the way of natural systems. Organization hierarchy is a poor replacement for connectivity. The power of Internet is in the fact that it allows the connection of many to many. That is why when it was one to many we could use a closed system, but with many to many we need to throw that system away.

Company J: I don't see anything that seems new to me—no difference from 13 years ago to now. Instead, the flashlight is in a different part of the darkness.

In three-person teams, you have vision, designers, architect, etc., in one room. In 15-person teams, you get problems with dialogue. Do you talk about them? Do you put them in a binder? How do you scale dialogue? How do you scale organization?

4.6 Plenary Discussion

Following the fishbowls, attendees participated in a general discussion to surface issues, differences, and assumptions underlying the fishbowl discussions and the study team's findings. Without reducing or constricting ideas, the group then identified those issues that they felt were of most significance and importance for the day's discussion. The result was a list of six outstanding issues—identified in the next section—to be tackled by breakout groups in the afternoon sessions.

The following is a breakdown of the observations, or "raw data," from the plenary discussion:

What is different? Collaborative technology allows purposeful connectivity.

- What exactly is different, specifically with agile methodologies: previously communication was often one to one; now communication can be one to many as well as many to many.
- Internet speed: Different from what?
- What is drastically new is that we have gone from a manufacturing-based to a knowledge-based economy. Manufacturing-based economies rely on transaction costs and differences in tools of production and place value on financial capital, while knowledge-based economies value human and social capital. They are very different. We need to look at the underlying environmental factors that led to this change. We are experiencing "punctuated equilibrium."
- What about "first principles"?

- One emerging trend is a return to "first principles" for good software development (XP collapses/flattens the life cycle, but the basic life cycle remains).
- A departure from "first principles" of software development is the distribution of power, which is significantly different. Distribution of power changes what is important.
- In complex systems, principles are discovered in simple rules embedded in each agent.

Another big difference is the difference in age. One of our "ahas" is that we work in one manner—late nights, more pizza and Coke. There is a way to do it better—to capture that energy in more repeatable ways. We are finding new ways to use old techniques. It's just that the Internet magnifies everything. What is it that Gen X brings to the table?

- Gen X has grown up on technology and is more interested in a trial-and-error approach.
- Gen X is willing to say, "I don't know." This is more difficult for Boomers.
- The dynamic of hoarding vs. sharing is changing. Gen X is willing to seek knowledge from the Web. Boomers are more likely to apply traditional methods where the reasons to do so no longer exist. The nature of the individual is changing the nature of the technology.

There are four additional areas of significant difference in Internet development:

- Privacy issues
- Accountability issues
- Intellectual property—ownership issues
- Security issues

The more things change, the more they stay the same. We say that implementation models are changing, but the exact same discussion appears 10, 20, and 30 years ago. The only substantial differences are the number and speed of connections, i.e., magnitudes of improvement in connectivity. Now when I put my things on the Web, I get a response three seconds later from China. Nothing is new here in a fundamental sense. What may be new is that more people are doing it: we have reached critical mass. Things done in the periphery are now reaching the core, because we have a critical mass of people doing it.

Software developers became strategists—like with the Yukon gold rush phenomenon. I need to be there, but I'm not sure what I'll do when I get there. A lot of companies arose due to some business idea, but I think there are still fundamentals that drive business. If developers are telling strategists what to do, there is something broken. There must be a reason and fundamental business drivers. Technology enables; it does not drive business.

Hierarchy: if we look back to what was different when I was a programmer, one key difference is the ability to communicate with peers, especially those in geographically different places. In the past, this level of connectivity wasn't there; however we are now lacking the rich interactions of the past. Now, Web-X exists, providing interaction without the sense of hierarchy.

Before we talked about object orientation and prototyping but nothing happened in practice. Now with XP and agile practices, change is really happening. Is it the "early majority" in Rogers' sense?

A few additional comments were made regarding the current state of software development at Internet speed:

- It may be possible to build quality software without having explicit requirements definition (e.g., eXtreme Programming only represents requirements implicitly).
- Functionality that crashes can still provide some good for the customer.
- In complex systems, simple rules are embedded in each agent.
- Business models and technology are changing at the same time.
- Are software architects an endangered species? eXtreme Programming can be interpreted as an emerging architecture model.

5 Six Burning Questions

From the plenary discussion, six questions of relevance and importance were identified:

1. What exactly is different regarding agile methodologies?
2. Internet speed: Different from what?
3. What is drastically new?
    - Shift from manufacturing-based to knowledge-based economy
    - Economy enabled by technology factors
        - transaction costs
        - tools production
        - value vs. human capital
4. Return to the first principles (e.g., good software development): what are they? How do they do this?
    - A departure from these first principles of software development is the distribution of power, which is significantly different.
    - In complex systems, principles are discovered in simple rules embedded in each agent.
5. Do we need requirements? Why write them down if they are always wrong?
6. Are software architects an endangered species?

Four groups were then formed and asked to select an overarching question or topic for discussion during the afternoon breakout session. The four topics derived from the six burning questions were:

1. Architecture issues in Internet software development
2. What is different about agile methodologies?
3. What is really new about the Internet environment and software development in that context?
4. Requirements in the Internet environment

6 Breakout Sessions

The goal of the breakout sessions was further exploration of the observations made in the morning session (a.k.a. the "burning questions"). To accomplish this, each group engaged in a process of hypothesis testing and difference questioning.

Hypothesis testing began with identification of an observation related to the topic. Then a set of hypotheses with potential to explain the observation was generated and tested. The purpose of this work was to begin the process of deep, thoughtful examination of an important trend or issue with Internet software development at the present time. The approach ensured that the groups avoided polarization in their views but continued to carry on fruitful dialogue, embracing differences as well as agreements.

Difference questioning flowed from the hypothesis testing and created a process for groups to develop a set of assumptions underlying each of their hypotheses. The groups then scanned their exploratory work to identify emerging principles and practices. The purpose of this exercise was to further build the community dialogue from generative ideas, and to ensure that the dialogue fairly represented all strongly held viewpoints. A further purpose was to allow for divergence and convergence of insights, with maximum cross-fertilization across differences.

Each group used worksheets to guide them through the process (see Appendix E). The objectives of hypothesis testing and difference questioning were:

- Brainstorm general observations about the topic.
- Formulate hypotheses surrounding the more enduring of the observations generated during the previous step.
- Articulate assumptions underlying some of the more insightful hypotheses generated.
- Express as meta-principles the immutable rules or axioms derived from the previous steps.

6.1 What is the Role of Architecture?

6.1.1 Target Question

The general charge to the architecture group was to consider the role of system architecture in high-speed software development environments. The group considered whether architecture was even required, whether it was predefined in the nature of the development, or whether it was key and central. Inevitably the group focused on the following question, driven by the consideration of whether anyone needed to pay any attention to the issue any longer:

> Are software architects an endangered species?

6.1.2 General Observations

The group began by accumulating a series of general observations. These represent the facts in high-speed development settings that the group members had experienced.

- There seems to be de-emphasis on software architecture in the high-speed development environment.
- Agile methodologies generally de-emphasize the need for architecture.
- The architecture of the computing infrastructure is separate and distinct from software architecture. This concept suggests that architecture may be maintained at a high level, but becomes unnecessary at a software level when high-speed approaches are used.
- In some instances, architecture can be divided into stable and volatile sections. This may not be a de-emphasis on architecture, but rather an isolation of the architecture issues for some parts of systems.
- In some cases, a standard, three-layer, taken-for-granted architecture has appeared. These layers are typically represented as database, business logic, and user interface.
- The Internet is a more stable environment in some key ways. For example, there is a more-or-less universal client (Web browser) available that provides a durable, standard user interface.
- High-speed systems development may be clustered around the business logic and user interface components. These components are more attuned to meshing with existing systems rather than creating the need for an independently developed architecture. For example, Web-based user interface components must fit into the existing architecture of legacy systems. It is unnecessary or not an option to develop an independent architecture.
- In these environments, architecture may be emergent and self-referential. The architecture evolves as necessary to meet the current needs of the rapidly changing system.
- The system architect may no longer be a single professional, but rather is a role that can be distributed or shared among all the developers. This is particularly the case in smaller groups.
- In some cases, the architecture of these systems is federated. That is to say, several different groups may own different parts of the architecture. There are also multiple views

of the overall architecture. Parts are shared by some groups, but are not referenced by other groups.

- Architecture is necessary to enforce coding standards for programmers.
- Many high-speed developments are smaller systems or components. Such systems may neglect the architecture. Architecture is really unnecessary for small systems.

From these observations, the group generally concluded that architecture was somehow less important in Internet-speed environments. But whether this was "good" or "bad" and why this might be the case were questions worth pursuing. The overall observation distilled for development in further detail was

> People put less emphasis than expected on architecture when developing at Internet speed.

6.1.3 Hypotheses and Discussion

The group proceeded to develop explanatory hypotheses that address the central observation above. These hypotheses focus on the question: "Why might people put less emphasis than expected on architecture when developing at Internet speed?" There were ten possible explanations suggested by the group:

H1: Methodologies in use don't put emphasis on architecture.

H2: Architecture is imposed by component assembly practices.

H3: Inexperienced developers don't know about architecture.

H4: Complexity of architecture overwhelms inexperienced developers.

H5: Architecture is perceived as expendable because it is not part of user experience.

H6: Developers believe architecture should be self-emergent.

H7: The available architecture implementation choices are constantly changing.

H8: There is a standard three-layer taken-for-granted architecture (that requires no treatment).

H9: Competent developers are unconsciously doing architecture without explicitly noting it.

H10: Architecture is actually expendable.

Once the group had enumerated these possible explanations, they proceeded to discuss and surface possible assumptions that were implicit in each explanation.

H1: Methodologies don't put emphasis on architecture. This explanation states that methodologies in general never treat architectural decisions. These decisions are not part of the methodologies. There are three notable assumptions: First, this explanation assumes that methodologies ought to treat and consider the architecture explicitly. An opposing view might hold that architecture should be determined outside of system development methodologies. Second, there is an assumption that developers should follow methodologies. An opposing view might hold that methodologies don't work. Third, there is an assumption that developers do follow methodologies. An opposing view might hold that developers never follow methodologies anyway.

H2: Architecture is imposed by component assembly practices. This explanation suggests that the development approaches that assemble systems from prefab software components absorb the architecture predefined by the component designers. There are four notable assumptions. First, the fact that methodologies do not emphasize architecture leads inexperienced developers to ignore architecture simply because they don't know anything about it. Second, components get designed with architectures in mind—the standard three-layer architecture concept, for example. This may mean that the concept of components and the concept of three-layer architectures reinforce each other. Third, this explanation assumes that components somehow have, imply, or define architectures. An opposing view might hold that components are generally architecture independent. Fourth, this explanation assumes that software development in Internet-speed environments involves components. An opposing view might suggest that these development practices don't use components anyway.

H3: Inexperienced developers don't know about architecture. This explanation suggests that untrained, brand-new developers haven't learned about architecture in school or on the job. They don't know it exists as an option or decision within their sphere of influence. There are four notable assumptions. First, this explanation assumes that Internet-speed developers lack experience. An opposing view might suggest that these developers are indeed experienced, and the explanation is nonsense. Second, the explanation assumes that inexperienced developers are the ones making architectural choices. An opposing view might hold that only senior, experienced developers make the architectural choices. Third, there is an assumption that teams do not have experienced developers to make architectural decisions (or at least spot the errors of their juniors). Fourth, the explanation assumes that it is possible to build a system without architecture. An opposing view might hold that all systems have at least an implied architecture.

H4: Complexity of architecture overwhelms inexperienced developers. This explanation states that the architecture issue is so complex and encompassing that the raw-recruit developers cannot cope with it as a problem. There are six notable assumptions, the first three of which are shared with H3. First, this explanation assumes that Internet-speed developers lack experience. An opposing view might suggest that these developers are indeed experienced, and the explanation is nonsense. Second, the explanation assumes that inexperienced developers are the ones making architectural choices. An opposing view might hold that only senior, experienced developers make the architectural choices. Third, there is an assumption that teams do not have experienced developers to make architectural decisions (or at least spot the errors of their juniors). Fourth, this explanation assumes that architecture choices are complex and beyond the ken of an inexperienced developer. An opposing view might suggest that architecture choices are intuitive and easily made. Fifth, it assumes that architectural complexity choices exist. An opposing view might hold that a complex architecture eliminates choices and developers are confined by the immutable architecture. Sixth, the architecture assumes that there is no "on-the-job learning." An opposing viewpoint would suggest that this kind of development environment quickly teaches developers about complex choices in architecture.

H5: Architecture is perceived as expendable because it is not part of user experience. This explanation suggests that users don't attach any value to architectural choices because they have no experience in differing architectures. Architecture becomes ignored in all of their design choices. There are eight notable assumptions, the first of which is shared with H3 and H4. First, this explanation assumes that Internet-speed developers lack experience. An opposing view might suggest that these developers are indeed experienced, and the explanation is nonsense. Second, it assumes that the ability to "scale" the system is not necessary or valued initially by developers. An opposing viewpoint would suggest the scale issues are apparent from the outset. Third, it assumes that architecture is mentally expendable, meaning that it is possible to ignore architectural decisions. An opposing viewpoint would suggest that it is impossible to build a system without thinking about its architecture at some level. Fourth, it assumes that architecture does not affect user experiences. An opposing view would suggest that users experience architecture, even an implied architecture, from their very first development work onward. Fifth, it assumes the possibility of one-tier applications—simple applications, self-taught and independent of architecture. Sixth, it assumes that user experience is more important than architecture, that user experience can indeed render architecture as perceptually expendable. An opposing viewpoint would hold that users cannot dismiss architecture on the basis of experience. Seventh, the explanation assumes that architecture is not verifiable. Therefore it can be hidden, and there is no accountability for architecture in systems. Eighth, it assumes that omitting architecture speeds development and possibly reduces quality. This is a questioned assumption by the group. But it suggests that strong, successful experience without an architecture leads to a conclusion that architecture is not only expendable, but also undesirable in Internet-speed development settings.

H6: Developers believe architecture should be self-emergent. This explanation states that developers choose not to make explicit choices about architecture because they think it should be permitted to change and evolve as the system grows and changes over time. There are two notable assumptions. First, that the self-emergence of architecture is a desired attribute of Internet systems. Second, the explanation assumes that it is possible for architectures to be self-emergent.

H7: The available architecture implementation choices are constantly changing. This hypothesis is a bit more tricky that the others. It suggests that the choices, the possible architectural configurations, are constantly changing. This changing setting means that architecture implementation choices are risky. A decision becomes a commitment. The implication is that developers "run away" from the risky commitment by never making a choice and never fastening on architecture. There are three assumptions worthy of discussion. First, it assumes that architecture is a costly choice. An opposing position would hold that architecture is cheaply made and cheaply changed. Second, it assumes that there is an advantage to choosing architectures later rather than sooner. An opposing assumption would suggest that early architectural decisions are advantageous. Third, it assumes that the architecture is really just a synonym for the technological platform. The technologies are changing and defining the architectural choices.

H8: The standard three-layer architecture is taken for granted and requires no treatment or consideration. This explanation suggests that these systems actually have architecture, one given and presumed. It is a perfectly suitable, pre-ordained architecture and therefore no decisions need be taken with regard to it. There are four notable assumptions. First, nothing particularly new has arisen in systems architecture, therefore there is no real reason for methodologies to address architectural decisions (therefore H1 is reinforced). Second, it suggests that Web-based applications are the only applications we really care about for Internet-speed development. This assumption implies, third, that Internet-speed development is addressed with three of four layers in architecture. The three layers are user interface, business logic, and middleware. The fourth layer, the database layer, is actually part of another (legacy) architecture.

H9: Competent developers are unconsciously doing architecture. This explanation suggests that developers are so thoroughly competent, either through training or experience, that their decisions regarding architecture are well-formed, but subconscious. This explanation assumes that there are actually competent, seasoned developers in Internet-speed projects. It may also assume that the three-layer architecture from H8 is really what enables this effect.

H10: Architecture is expendable. This explanation suggests that architecture really doesn't matter, at least for these kinds of systems. Therefore it is not necessary to make architectural choices. (The group did not discuss any further assumptions.)

6.1.4 Meta-Principles and Discussion

The group then went on to develop six principles from this set of explanations and assumptions. The principles are:

P1: If you have inexperienced developers you should choose a safe, tried-and-true architecture.

P2: There is more than one reason for the de-emphasis of architecture.

P3: Different modes of Internet Development require different kinds of architectures.

P4: Effects of bad architecture (may) show up too far downstream, and only if costly problems occur.

P5: Throwaway systems need no explicit architecture.

P6: Internet development is two-phase: architecture-free and architecture-bound.

In discussing these principles, the question arose as to whether there was any experience in which a system was known to have failed due to bad architecture. The group discussed several war stories, but the best example was that of E*TRADE. At one point, E*TRADE was down for two days, and this failure was clearly laid to blame on a poor architecture.

P1: If you have inexperienced developers you should choose a safe, tried-and-true architecture. This principle is based on the belief that developers are not trained in architectural choices. It is not clear whether this is an oversight in the general training of developers, or simply that this kind of design decision really needs to be based on the experience of practice. Therefore inexperienced developers are really not capable of making architectural decisions, and a well-known architecture should be selected. There is also the issue that inexperienced developers may perform best in safe architectures. These systems are likely to be more robust given developer errors in software design and implementation. What's more, these safe architectures limit the choices of the inexperienced developers in terms of their software design and implementation decisions. Therefore there are fewer erroneous pathways for them to take. This limitation is important when inexperienced developers have to make decisions rapidly. Safe architectures are also more likely to be familiar to developers in the sense that their training may have taken place under similar architectural decisions.

This principle originally read "There is a belief that in an environment of rapid change and inexperienced developers you can limit the number of choices by making the architecture fixed." It was later revised to read as above.

P2: There is more than one reason for the de-emphasis of architecture. It is clear that architecture has assumed a background role in Internet-speed development. There is no simple reason for this. It is a multifaceted and complex event. Simply reintroducing an emphasis on architecture is not necessarily going to improve Internet-speed development.

P3: Different modes of Internet-speed development require different kinds of architectures. For example, in some settings there are very limited resources, very poor experience, and fairly small scale issues. This development could operate in a significantly different mode than one in which there is a large-scale task with good resources and good experience. The architectural choices in these two different modes can also be quite different, and each mode, with the correct architecture, can succeed within the limit of its goals.

P4: Effects of bad architecture (may) show up too far downstream, and only if costly problems occur. This principle suggests that in some circumstances it can be a wise decision not to choose an expensive architecture. Such architecture might enable robust growth and adaptivity in the system. But it might also slow development and increase the cost. The damage done by such a bad architecture is not likely to surface until well into the system life span. If the setting is one in which the system may not be long enduring, then the effects of the bad architecture may never have enough time to materialize.

P5: Throwaway systems need no explicit architecture. Along the lines of P4, if a system is being built as a throwaway prototype or as a temporary problem fix, then a good architecture may be an unnecessary expense.

P6: Internet development is two-phase: architecture-free and architecture-bound. The issue of architecture in Internet-speed development might be best viewed as a two-stage decision set. In the first stage, "a market capture phase," developers can choose to de-emphasize architecture. That is, the first stage entails building a throwaway system that may lack robustness, maintainability, scalability, etc. However, if the market fails to sustain the software product, then no undue investment in system architecture has been risked or lost. In the second stage, "a market sustainable phase," developers can choose to emphasize architecture. That is, the second stage entails rebuilding the original system using a more robust architecture. In this stage, the market has justified the risk of investment for the better-built system. The development pace, methodology, and practices will probably be radically different between these two different stages or phases.

Implied here is the idea that one form of development is used for risky, low-demand settings. If the market for the developed system justifies continuance of the system, it is rebuilt. On the other hand, if the product flops in the market, the loss is minimal. The group discussed the tendency for small firms funded by venture capital to develop Internet-speed software products until the market has validated their viability. At this point the product is sold to a larger firm that has the capital basis for redeveloping the software product using a proper architecture as a long life, robust system.

6.1.5 Conclusions and Promising Practices

The group finally discussed promising practices that rest on these principles. eBay was discussed but it was never resolved whether their experience reinforced or conflicted with the principles. They discussed the E*TRADE outage that resulted from their bad architecture, and the way in which Charles Schwab took advantage of this outage. The implication was that the second-stage redevelopment must not be postponed too far. There was also a suggestion that development environments should also be staged in terms of complexity. For example, use Microsoft ASP for the first stage. Postpone more robust Java development until stage two.

Company B was discussed as an exemplar for managing the two-stage development modes. Their early development was in an unstable context. Their future was unclear. They pushed architecture worries out to the horizon until their future began to stabilize.

The group also noted that all of this Internet-speed development technology is so new, both in development and the environment. It seemed reasonable to partner with a technology provider to mitigate risk.

6.2 What is Different About Agile Methodologies?

6.2.1 Target Questions

The objective of this group's discussion was to analyze whether there is anything different about agile methodologies, vis-à-vis traditional software development methodologies. The main questions posed by the group included the following:

- What agile methodologies are being used in software development?
- What distinguishes these agile methodologies from traditional software development methodologies?
- Are agile methodologies *really* different?

6.2.2 General Observations

To analyze these questions, the group members began by making a number of general observations about agile methodologies.

- People are claiming that agile methodologies are the best way to do information systems development. Agile methodologies have emerged as a "new and radically different methodology." One can pick up any software magazine and find that claim.
- Three key elements of agile methodologies include collaborative work, incremental evolutionary life cycles within a strategy, and strong customer communication.
- One way of approaching the questions is to go through the basic practices of eXtreme Programming and other agile practices. Note that XP embodies most of what is good about agile methodologies.

  The key practices of XP include

  - valuing individuals and interactions over processes and tools
  - working software over comprehensive documentation
  - collaborating with the customer over contract negotiation
  - responding to change over following the plan

- Agile methodologies reward heroes.
- The effectiveness of agile methodologies appears to lie in people, *not* process.
- People are ignorant of good software engineering practices. When people see something they don't like, they deviate to the other end of the spectrum (to illustrate: if people hate the life-cycle process, they rationalize "who needs this anyway.").

6.2.3 Hypotheses and Discussion

The group generated a number of different hypotheses and underlying assumptions about agile methodologies and their role in software development.

H1: Agile methodologies are more effective (than traditional software development methods).

H2: Agile methodologies are not different from traditional software development methods.

H3: Partially implementing key agile practices will lead to project failure.

H4: Agile methodologies require good people to be successful.

H5: Internet-speed development is fundamentally different from traditional development; thus, agile methodologies are needed.

H6: Agile methodologies are effective when the time horizon is short term, and not as effective over the long term.

H7: Agile methodologies aren't really new per se, but their implementation is extreme.

H1: Agile methodologies are more effective (than traditional software development methods). There is absolutely no empirical evidence to support this claim (e.g., there are no studies comparing the effectiveness of parallel programming and XP). Early anecdotal evidence suggests that agile methodologies projects get finished early. Perhaps we can't (shouldn't) ask for more from a methodology that is not mature.

H2: Agile methodologies are not different from traditional software development methods. The principles that agile methodologies follow are not different from those of traditional methods, but it is the implementation that is different. (Note that this is a key insight of the group, and members return to this insight later in the discussion.)

H3: Partially implementing key agile practices will lead to project failure. Software development requires a balanced approach. If you do only one side of the suggested agile practices, then that will lead to failure. For example, *Built to Last: Successful Habits of Visionary Companies* [Collins 97] talks about companies that have been successful over the decades, and should be an influential book for information systems development. Authors Collins and Porras find that what is special about visionary companies that are able to last is that they have a "core ideology" or identity and they actively indoctrinate employees into ideological commitment to the company. The book demonstrates the benefits of innovation of the market leaders. Visionary companies have an atmosphere in which bright people are keen to see the company succeed and unafraid to try a lot of stuff and keep what works. Many of the visionary companies are characterized by lack of an initial business master plan, little structure, and no prima donnas.

H4: Agile methodologies require good people to be successful. The only people who can use agile methodologies successfully are skilled, experienced software engineers.

H5: Internet-speed development is fundamentally different from traditional development; thus, agile methodologies are needed. The focus in Internet-speed development is internal versus external collaboration.

H6: Agile methodologies are effective when the time horizon is short term, and not as effective over the long term. (1) Don't try to plan too far ahead. "Go as far as you can see, in the general direction you need to go." A metaphor to illustrate is that of an explorer trying to find a path through the Sierra Mountains without a map. (2) There are questions about the feasibility of performing long-term projects with agile methodologies (XP in particular).

H7: Agile methodologies aren't really new per se, but their implementation is extreme. For example, Egis has an architecture and design, which is effectively de-emphasized in XP. XP emphasizes features. Refactoring occurs at the code level rather than at the architecture level. This may be appropriate for a given business environment. Architecture is not explicitly tackled in XP. It is believed to be emergent.

6.2.4 Key Insights

The group then identified a number of key insights. These include the following:

- The basic principles of software engineering exist, are known, and are immutable.
- There is more than one way to implement a software development principle.
- Some combinations of practices/implementations are superior in specific environments. This means that different environments require different implementations for effectiveness.
- For every agile methodologies practice, an analog in traditional software development methods can be found. For example, consider the following principles. These hold in both agile and traditional methods for software development:
  - Good teamwork is critical.
  - Requirements change.
  - Talking to the customer is good.

- Although everyone agrees on software development principles, some practices are controversial.
- Software development is linear and time bounded at an atomic level. Consideration of the way you compose the life cycle over time changes the way that you attack the problem.
- The choice of software development life cycle dictates the choice of practices such as waterfall, spiral, etc.
- It's important to describe what the principles are and how the practices are different from one methodology to another.
- Environments dictate practices, not principles. You can't apply the practices from a stealth bomber software development project to a commercial environment. It is a different environment.
- Project size affects software engineering practices.

6.2.5 Meta-Principles

From these insights, the group members then generated the meta-principles underlying software development and software development methodologies. The meta-principles they defined are as follows:

P1: All software processes require good teamwork regardless of methodology.

P2: All methodologies have to accommodate changes to requirements.

P3: All (successful) methods derive from the basic principles of software development.

P4: Good software development methodologies engage the customer.

P5: There is more than one way to skin a cat.

P6: The project environment constrains what software development practices will be effective.

P7: Practices often interact in unforeseen ways. The implication is that you can mix and match methodologies, but sometimes with unintended consequences.

P8: Current agile methodologies are a simple instantiation of a set of practices that seem to work.

6.2.6 Promising Practices

Finally, the group concluded the exercise by identifying what the members saw as promising practices in agile methodologies. These include

- Short development iterations. This allows developers to explore the unknown problem space effectively without wasting a lot of time.
- Accountability. There is a difference between hackers and XP: discipline ("do what you say and say what you do"). If programmers aren't disciplined, they aren't agile, they're hackers.

- Minimal essential documentation. One group member noted: "I've done it in my company and it works." Documentation assumes we'll exist to need it. Another member noted: "Documentation doesn't have to be formal, but it has to be there. We have a form where we write down the requirements and design. We capture the form, so that the next guy in the team can pick it up."
- On-site customer for technical customers. But note that having customers on site raises a different set of problems.
- Customer liaison/engineer at customer site
- Pair programming
- Frequent peer reviews
- Frequent customer reviews. What is the best time for this to happen? One group member noted that normally it happens when we come to a stop in the cycle. Often it results from "golf course commitments" between managers and customers without involvement of the engineering staff.
- Frequent regression testing
- Configuration management. Not explicitly mentioned in XP; however, may be implicitly addressed or assumed to be in place.
- Detailed short-run planning, fuzzy long-run planning. Methods limit the type of planning that is feasible. The implication is that you can plan in detail for three to six months, but not much more than that. Some say that in a rapid development environment you should have detailed planning one to two years out. This seems unlikely.

6.2.7 Conclusions

In response to the questions analyzed, a major finding of this group is that agile methodologies may appear to be different in implementation, but are not really different in terms of software development principles. Also, while software development principles don't change, different implementations (or practices) may be more effective in different environments: there is no "one size fits all" approach that is best for all projects.

6.3 What is New About the Software Development Environment?

6.3.1 Target Question

This group chose to focus on environmental factors affecting the development of current software. Because of the breadth of the question, the group chose to focus in on two specific environmental factors.

"What is drastically new about development today?"

- Shift from manufacturing-based to knowledge-based economy

- Economy enabled by technology factors
  - transaction costs
  - tools production
  - value vs. human capital

6.3.2 General Observations

The group made several observations regarding current software development practices based on the members' experiences.

- Manufacturing vs. knowledge sharing paradigms, etc.
- Information is networked, not broadcast as much. Not filtered.
- Manufacturing assumed that we had full information rationality and we could use deductive reasoning; however, knowledge economy emphasizes partial information, induction, and intuition.
- Huge amounts of information are readily available to everyone on every level of the organization. There's no stopping it. You limit yourself (and your organization's potential) by filtering information.
- We used to see people acting and interacting as a series of steps, in a sequence. Now we see people interacting as a community for a particular result.
- Management structures are looking for fixed, standard, hierarchical, efficiency. Real work is fluid, cyclical, and emergent.

6.3.3 Hypotheses and Discussion

The group's development of explanatory hypotheses centered upon tensions between a manufacturing economy and a knowledge-based economy with respect to the current software development environment.

H1: Management came up through the ranks of assembly lines.

H2. For historical reasons, organizations are enamored with machines.

H3. People like the sense of control, prediction, cause and effect.

H4. Measures are oriented to efficiency (productivity) and not creativity or innovation.

H5. The assembly line emerged in a different educational/cultural context.

H6. Fear reinforces old management models.

H7. The tools of production have changed.

H8. There is a potential re-emergence of the guild system of training.

H1: Management came up through the ranks of assembly lines. Management represents an older generation. This generation has been entrenched/vested in assembly line mental models: they are trapped by their experience. Management has most likely been educated via traditional business paths of the mid-century, indoctrinated by academia and older consulting firms with vested interest in perpetuating the doctrine, and therefore seems out of touch to its Gen X subordinates.

H2: For historical reasons, organizations are enamored by machines. This notion that machines can make things bigger, better, faster, comes from a time when efficiency was king. Additionally, the old workforce was mostly male, and males were historically enamored with machines.

Although the workforce has changed, the old rules still dominate because management is still indoctrinated in the old mentality. Under their doctrine, "efficiency is good" and "messiness is not good," and efficiency is still associated with the machine/assembly line mentality.

H3: People like the sense of control, prediction, cause and effect. Uncertainty leads people to believe that they are losing control. Expectations drive the level of discomfort, which in turn drives stress levels as things spiral out of control. We don't trust each other's judgment or view of things and become panicked as our uncertainty grows.

H4: Measures are oriented to efficiency (productivity) and not creativity or innovation. Measures of efficiency really reflect levels of efficiency.

H5: The assembly line emerged in a different educational/cultural context. The production was centralized and expensive and down time was not an option because it was cost prohibitive.

H6: Fear reinforces old management models. People in power want to keep it, and fear losing it. There are no alternative tools, techniques, and education that seem viable to people who fear that they might be replaced.

H7: The tools of production have changed. The tools of production are now distributed and available at a lower cost.

H8: There is a potential re-emergence of the guild system of training. The guild system went away because of the introduction of the assembly line, but could return because of the increased availability of the necessary tools of production (by guild system we mean masters and apprentices). Now, multiple masters are possible anywhere. In the master-apprentice model, there is tacit and explicit knowledge, as well as explicit rules to be followed. Technology is enlarging and enabling the master-apprentice concept. Social and intellectual capital is now valued over physical production.

6.3.4 Meta-Principles

The group developed three principles that they felt were driving current software development practices, after reviewing their hypotheses and assumptions.

P1: Your incentives keep things the same (tenure, consulting approaches/fees).

P2: Any time your expectations are compromised and stress is created, disconnection between expectations and reality produces more stress.

- H = R - E: happiness equals reality minus expectation
- models expectation

P3: Tenure perpetuates the status quo; offers incentives to keep things the same.

6.3.5 Conclusions

The group discussed the observations and hypotheses and consistently returned to acknowledge a real and existing tension between the old manufacturing economy and the new knowledge-based economy.

While knowledge sharing is embraced as laudable, inevitable, and necessary, organizations remain largely entrenched in the values, measures, and structures of an outdated manufacturing paradigm.

The university educational system, in the groups' view, was not seen as an accelerator for change; rather, the tenure system was seen to perpetuate the status quo. In the long term, a shift toward knowledge-based theory or practice was seen as inevitable. One of the most exciting future opportunities included a reinvention and re-emergence of the guild system and the master-apprentice model. This time, however, as a result of knowledge networks, multiple masters would be possible anywhere.

6.4 What is the Role of Fluid Requirements?

6.4.1 Target Question

The objective of this group was to analyze the role of fluid and evolving requirements in high-speed software development for the Internet. The main questions posed by the group included the following:

- What is different about the nature of requirements for Internet software development?
- Is it possible to build software without explicit requirements definition, say using requirements that are implicit or emergent?

To analyze these questions, the group made several observations about the nature of software that is developed in the Internet environment, the process of systems development, and the role of requirements for building systems in the Internet environment. Then the group generated several hypotheses concerning how requirements get identified, evolve, and are managed in high-speed software development. The group then examined assumptions behind these hypotheses. This was followed by the identification of meta-principles that guide software development in this environment.

6.4.2 General Observations

At a fundamental level, software now plays the role of an enabler of the economy. The impact of technology is greater with more complex requirements. Therefore software becomes the business. This fact is the basis for the following observations:

- There is a general need for speed due to competitive pressures. First to market or early to market is a requirement in most Internet businesses.
- The definition of quality is changing and malleable. To accommodate the need for speed, customers and developers are willing to sacrifice certain aspects of quality while developing systems. The requirements for quality are not cast in stone. They are quite malleable and change according the current market demands for quality.
- The importance of security requirements has increased. As the use of Internet applications has increased, especially with critical business systems getting exposed to users outside the organization, the need for security has increased.
- When we build simple applications, the requirements for them can be well specified. As we build more and more complex software, it is not possible to fully articulate requirements.
- In Internet software development, often users don't know what they want. Many of the systems are creating new products, services, and markets. The requirements for these are not fully understood by anyone.
- Rather than relying on formal specifications, users and sponsors rely on informal dialog to convey requirements.
- Distance is compressed because tools are available to do work wherever you happen to be. Collaboration tools can be used to facilitate the dialog about requirements among the various stakeholders.
- Creative aspects of software are becoming very important, but this aspect of software requirements is very difficult to get right.
- Domain experts may not exist for "new economy" applications or ways of doing business. Therefore, it is difficult, if not impossible, to fully specify requirements for such projects at the outset.

6.4.3 Hypotheses and Discussion

The group generated a number of different hypotheses about the role of requirements in high-speed software development. Assumptions underlying the hypotheses were also discussed.

H1: Internet software is used to support new business models for which requirements cannot be articulated clearly in advance.

H2: The accelerated pace of development does not allow for lengthy requirements processes.

H3: Interface requirements are critical; perhaps they are over-emphasized or their importance is disproportionate.

H4: Requirements definition, rather than requirements specification, is feasible in Internet software development.

H5: Written requirements do not equal correct requirements.

H6: Documentation is not the end product.

H7: Internet software development requires more direct coupling of users and systems designers.

H8: Maintenance is not critical.

H1: Internet software is used to support new business models for which requirements cannot be articulated clearly in advance. Internet software is often used to create new businesses that didn't exist before. New products and services are created in this space. Software development starts with a very sketchy understanding of what these products and services are and the details get refined and "discovered" as the development continues. Even the customers may not fully understand the scope and potential of these offerings until they begin to experiment with the early versions. Software developers have to start with a sketchy concept proposal and develop a concrete realization of it. When expertise does not exist or existing expertise cannot be easily extended, complete requirements for the software cannot be well specified at the outset (i.e., you can't have fully specified requirements for a business that no one understands). In addition, competitive pressures force innovation in directions not originally planned. Therefore, requirements may change, sometimes dramatically, during development.

H2: The accelerated pace of development does not allow for lengthy requirements processes. The pace at which businesses operate, especially in the Internet space, has accelerated. Solutions need to be developed, deployed, and evolved rapidly, matching the speed with which the underlying business context operates. In many software development activities, time to market is considered the prime driver. To meet this demand for speed, software development processes have to be adapted and accelerated. Rather than relying on lengthy requirements specification processes, many organizations use an iterative process for understanding evolving requirements.

In some areas, technology has evolved too rapidly for businesses to react. However, due to competitive pressures, organizations are forced to develop and deploy systems. In this scenario also, development proceeds at an accelerated speed, giving very little time for a lengthy requirements engineering process. The uncertain nature of requirements also requires an iterative approach.

H3: Interface requirements are critical; perhaps they are over-emphasized or their importance is disproportionate. Systems deployed on the Internet have potentially millions of users in many cases. When dealing with such a large number of users, the design of the user interface becomes extremely important to make the system successful. Unfortunately, the number of potential users also makes the design of the user interface extremely difficult. Ascertaining user requirements for the interface to satisfy a variety of user segments with very different backgrounds, skills, regions, countries, languages, and cultural backgrounds is exceedingly hard. A significant proportion of software development effort is devoted to the development and customization of user interfaces.

However, software systems geared towards providing infrastructure services and performing backend processes do not have to deal with the complexities of developing user-friendly interfaces.

H4: Requirements definition, rather than requirements specification, is feasible in Internet software development. Requirements definition differs from requirements specifications in terms of their formality. In high-speed development, precise specification of requirements is often infeasible, as the requirements are often ambiguous, incomplete, and constantly evolving. Therefore, rather than trying to achieve rigorous specifications, developers are willing to work with informal requirements statements. These may be specified as scenarios describing the use of the system under different conditions. As users work in close proximity to the developers, clarifications of these requirements can be easily obtained from them as the development proceeds.

H5: Written requirements do not equal correct requirements. Written requirements are frequently misunderstood to be correct requirements. Rather than give a false sense of completeness and finality to the requirements by formally writing them (in a requirements specification document), many organizations rely on informal mechanisms for understanding user requirements. These may include Joint Application Development (JAD) sessions, co-locating the users with the developers to facilitate quick and informal feedback. Also, prototypes are used to get feedback from users. This approach also helps in changing priorities for development mid-stream.

H6: Documentation is not the end product. Understanding and satisfying requirements, rather than creating documentation, is the goal of the requirements engineering process. High-speed software development can benefit from a people-centric rather than a documentcentric approach. This is especially true for the requirements engineering processes, for the reasons mentioned above.

H7: Internet software development requires more direct coupling of users and systems designers. Unlike traditional, large-scale systems development, Internet software development is done with close collaboration with the users. First, many applications are initiated by the users. Many domain experts are also technology savvy. The technology for development, at least for prototype development, has become simple and user friendly. Some users may even have developed the first prototypes. As these users are more experienced with the types of applications they want developed, they can actively participate in shaping the direction of development. Even when the users are not directly involved in development, they are often co-located and work in close contact with the developers. Having the ability to shape the development, sometimes in unplanned directions in response to changing needs and market pressures, makes the users feel empowered.

Another interesting trend is that technologists who understand the domain well initiate many projects. This combination of technical and business skills is rare in traditional systems development.

H8: Maintenance is not critical. Internet software development is frequently done with the assumption that the life span of software is very short. First, the deployed software can be replaced with newer versions. The need for speed may not provide the developers sufficient time to develop an appropriate architecture for the system to be evolved from version to version. Therefore, it may be faster and cheaper to re-implement successive versions rather than maintain them. Also, the nature of the application may change so dramatically over a short span of time that the software cannot be modified to meet current needs.

6.4.4 Meta-Principles and Discussion

From the discussion of hypotheses and assumptions, the group generated the following meta-principles governing Internet software development.

P1: Sources of change are a constant. Change impacts all facets of the operating environment, including technology, business, economy, people, and competition.

P2: We can't escape the four underlying principles of software development (or business).

P3: Focus on communication rather than documentation and specification.

P1: Sources of change are a constant. Change impacts all facets of the operating environment, including technology, business, economy, people, and competition. A recurring theme during the entire discussion was that the only thing that is constant is change. Change happens in every facet affecting Internet software development. The business climate, opportunities, and threats are changing rapidly. The technology for software development is evolving very quickly. There are tremendous changes in the skill sets, expectations, and roles of software developers, as well as users. Internet software development practices have to accommodate all these changes while maintaining a balance with the need for quality.

P2: We can't escape the four underlying principles of software development (or business):

- requirements of some type
- how to build whatever is required
- actually build it
- test and make sure it works (satisfies the requirements)

The basic functions in software development (analysis, design, and implementation) have not changed with Internet software development. However, the practices used in these activities have to be tailored to the specific needs of high-speed development. For example, rather than a sequential waterfall model, a spiral or iterative development methodology is needed.

P3: Focus on communication rather than documentation and specification. The purpose of requirements engineering processes is the understanding of user requirements in sufficient detail and clarity that they can be realized in the software system. This can be achieved by adopting practices that foster open and frequent communication between the users and the developers. Focusing on documentation of requirements is ineffective when dealing with rapidly changing requirements in response to changes in the environment. Also, Internet software development often involves creating new products and services whose requirements have not been fully understood. In such cases, it is futile to attempt to formalize requirements in a specification document.

6.4.5 Promising Practices

Finally, the group concluded the session by identifying what the members saw as promising practices in dealing with fragile requirements. These include

- using Joint Application Development sessions to identify, refine, and evolve requirements
- using focus groups to identify user requirements and solicit feedback on early prototypes
- co-locating users with developers so that the physical barriers to communication are reduced. This helps developers obtain immediate response to their questions and lets the users drive the direction of development efforts.
- getting frequent feedback on prototypes from users

- incremental development of functionality in successive releases so that customers can dynamically prioritize the features that need to be included in successive versions of the product
- informal specification of user requirements as scenarios rather than relying on formal specifications
- direct user involvement in prototype development, when working with customers who are savvy in development methods and tools

7 "Futuring" & Scenario Development

7.1 Brainstorming Session: Emerging Conditions and IT Impact

The purpose of this exercise was to extend the identification of several key issues in the emerging world of Internet speed and software development, and to create products of value to the participants and to the study team in their continuing work with Internet software development.

Participants, as a full group, identified emerging and probable future conditions and their potential impacts. Because of the time constraints, two specific scenarios were selected for the group to work on. These scenarios are intended as examples, and the findings are incomplete.

Note: There is some overlap between the two lists but they should be considered independent of each other. Although they are linked, conditions are not lined up with their specific impacts.

Table 5: Emerging Conditions and IT Impact

Emerging Conditions	**Possible Impact /Implications on IT**
Globalization	Take the foundation of product and put cultural interpretations to interface
Disparity between the haves and have-nots. The digital divide will increase globally.	What is a luxury in one place is a necessity in other place?
Distributed global collaboration	What is the fair price of software development in developing countries?
Ever-increasing affordable bandwidth	More and more distributed development
Dot.coms will face the fact that they have to maintain their products	Increased maintenance costs and staff
Health care arena will continue to grow	Healthcare focus, demands on quality are going to change because of life or death emphasis.
IT worker compensation	Incentives for software developers. Gen X and Baby Boomers, and recent events—are stock options effective? If not, what is going to take their place?
Fundamental changes in software training will occur	
Move toward open systems and more wide-open distributed development	

Emerging Conditions	Possible Impact /Implications on IT
Changes to business travel patterns	
Changes in consumer patterns. Move towards Internet and online shopping.	
Outsourcing will become more common and facilitated by the Web	Same infrastructure opens up possibilities in out-sourcing
Concerns over worldwide terrorism will shape future for fore-seeable future	We will see companies working with other compa-nies, a lot driven by security
Customized versus COTS software. COTS will dominate.	How much software development will be done on customized ___?, software development is still ex-pensive—moves toward component development
Customer expectations—more privacy and functionality	
Change in quality definition, quality is negotiable	MS.net is moving to server side model. So user interface will suffer.
Shrink-wrap software becomes business commodity	
Tele-presence more flexible and viable alternative for many employees	
More inter-organizational collaboration	
Web access continues to grow and reach further into society	
System interoperability becomes critical	
Strength of the open source movement increases and becomes more important	Would like to see open source grow, but in light of new changes, expect that to shrink, move to larger platforms Uncertainties about intellectual property protection for software
Changes in the definition of information workers or employee	Team concepts are up for grabs. Although people recognize the need for teams, one team member is more equal than the other. With Internet, and dis-tributed collaboration, what we have is transparency of value. We know Michael Jordan bring crowds to teams. I want customized motivation that fits my value, based on who brings what to the software development. You know who the good people are.

7.2 Force Field Analysis

What is a likely and significant area of impact for the future: two scenarios.

7.2.1 Scenario 1

Security holds life and death of the Internet

Enabling factors: **Factors that might facilitate this scenario**	**Constraining factors:** **Factors that might mitigate this scenario**
We can solve the security problems, but that destroys what is good about the Internet	People will reinvent the Internet: Internet II
Easy cyber-terrorism target due to poor security archi-tecture	Security highly developed

Enabling factors: **Factors that might facilitate this scenario**	**Constraining factors:** **Factors that might mitigate this scenario**
Wintel as Internet standard platform fundamentally cannot be secured	Security neutral aspects of Internet are ubiquitous and entrenched in corporations
Widespread event that destroys the trust in the Internet (e.g., World Trade Center-type attack on the Internet)	Change/rescue (in case of event of dramatic failure)
Health security	
National infrastructure on the Internet will put more restrictions on the Internet. Government get more involved in regulating security.	Moore's law
Security experts are always reactive rather than proactive. They can't keep up with the "black hats."	More government regulation
	Internet is already in wide development and use
	Much more work is done on the Internet than has been done ever before. There's no going back now.
	Non-secure aspects of the Internet are robust, or irreversible. Embedded into the culture.
	We'll accept more risk and inconvenience in exchange for access.

7.2.2 Scenario 2:

Individual developers will be free agents; end of employees

Enabling Factors: **Factors that might facilitate this scenario**	**Constraining Factors:** **Factors that might mitigate this scenario**
Younger generation mindset: growing up trial and error, self-teaching, not feeling the depression or the need to be buffered. Now there are no buffers. There is no loyalty.	Difficult for most people to survive economically on their own
Safer to be an independent than an employee of a company	Large company can pay people to do development in R&D/innovation, but people on their own can't afford to do it
As an employee, I'm loyal to company until I get my pay	Most engineers and programmers find the notion to operate as an independent not an attractive one. They like to program, don't like to sell themselves. They don't like people. Interpersonally challenged. (Part of the job is always social. But it is another level.)
I am loyal to my peers	But the model only applies to Michael Jordans. How many of these are there?
Learning among peers, continuous in environment. Learning is distributed.	What motivates software engineers? Learn from others, work with company that thinks about my training. But they are not good at selling themselves.
But if transparency of value captures your value without having to sell yourself, then you don't have the problem of selling yourself	Intellectual property issue (both a + and -). People don't know how to deal with it.
People want to work with good people	Transaction costs lower within company than across companies. Ease with which you can get information, collaborate with other.
Outsourcing is pushing the issue	Continuing education is hard for individuals to do

Enabling Factors: **Factors that might facilitate this scenario**	**Constraining Factors:** **Factors that might mitigate this scenario**
For those projects that we find easy to communicate, we should outsource. If it involves complex development, don't outsource because it will cost you more.	Lack of portability of benefits and uncertainty of income
Corporate memory more rapidly obsolete	People are interested in fostering and building corporate memory
	Most interesting projects are still inside company projects. Free agents don't get such opportunities.

8 Conclusions

8.1 Synthesizing the Learning

Specific findings from the colloquium include all the information summarized in this report: observations made during the active listening circles, results from the four breakout groups' discussions, as well as the output from the plenary group's activity on scenario development. In addition, soon after the event, the study team took time to recapitulate and synthesize additional valuable insights that had been gained from its perspective—that of a research team conducting a longitudinal case study and investigation of Internet speed and fast-paced development. The following topic areas and items were earmarked as significant. The research team continues to investigate these areas.

Traditional versus New Approaches

- There are two models of software development: the old manufacturing economy model, and the new knowledge economy model. There is limited support for knowledge-oriented operations at this time because few, if any, management practices or structures have been created. These are needed to overcome the status quo in development.
- More traditional approaches to systems development are founded on a philosophy of a rational cleanup of the messy world. Internet-speed development is different because it operates with a philosophy in which the world is left in its messy state: we work in the mess without trying to straighten it up. This is contingency theory in a "big way."
- Many fundamental, traditional ideas are finally giving way to new ways of thinking. The population of "new think" believers has finally grown to a critical mass necessary to precipitate a revolution.
- There is a palpable generation gap in the software development community: Gen X versus Baby Boomers.

What Changes, What Endures

- So many aspects raised by Internet-speed development are transitory, and indeed everything may have become transitory in such a way that the transition will now be continuous and will never complete.
- Principles don't change…but practices have evolved. Implementation of principles depends on the environment.
- Durability is defined by changeability…that which endures is changeable.
- One of the real changes in systems development has been the connectivity of the world, and more importantly, the systems developers.

Products, Processes and Markets

- The user interface with the customer is much more important to Internet-speed development than in the older, more traditional development styles.
- Agile methodologies do not treat configuration management (maybe it is assumed).
- Pricing of software development is being affected by differential international development (outsourcing appropriate fragments to low-cost venues).
- Systems are being (or ought to be) constructed in two ways: "built-to-last" versus "built-to-capture [market]."

Appendix A Colloquium Participant Invitation

Software Development @Internet Speed

Does Speed Kill?

"Of all the new economy's supposed 'rules,' the notion that nothing is as important as being first to reach scale may be the most widely accepted. It's also wrong."
– Jim Collins, Good to Great

"Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."
– The Agile Manifesto, signed by 17 leading software developers, The Agile Software Development Alliance, February 2001

"If the 1980s were about quality, and the 1990s were about reengineering, then the 2000s will be about velocity. About how quickly the nature of business will change... A digital nervous system will let you do business at the speed of thought—the key to success in the twenty-first century."
– Bill Gates, Business @ The Speed of Thought

You are invited to participate, as a selected contributor, in a Discovery Colloquium focused on one of the most critical issues in software development today:

* * * * * *

"Developing Quality Software at Internet Speed"

When: October 18, 2001
Where: Software Engineering Institute (SEI)
4500 5th Avenue, Pittsburgh, PA
Sponsors: A joint study team from SEI, Carnegie Mellon University, and Georgia State University

* * * * * *

You have been selected because you have important things to say about Internet-speed development. The Discovery Colloquium will explore compelling tensions, tradeoffs, and future forecasting in the realm of fast-paced development. The Colloquium will depart from the usual conference format of speakers, panels, and workshops. In keeping with the emergent nature of the Internet-speed environment and the expected vitality of exchange among all participants, the Colloquium will draw on innovative methods to encourage open dialogue, creative generation of ideas, and the richest possible outcomes.

The Colloquium will spotlight and encourage divergent views—not to drive wedges but to build bridges of communication and expanded knowledge. The goals are to craft a vision for the future of software development, and to identify promising trends, practices, and approaches for next-generation software development.

Participants will be drawn from diverse areas:

- Entrepreneurial and traditional brick and mortar companies
- Proponents of agile methodologies, model-based & process-based methods
- Internet strategists and researchers
- Software developers and managers

In the interests of supporting this critical discussion, Colloquium sponsors secured funding to underwrite the Discovery Colloquium to make it possible for every invited participant that would like to attend to be able to do so. Registration fees for the Colloquium are nominal, $150 intended to cover basic costs for the event only. See separate forms for information on registration and housing.

Appendix B Colloquium Brochure

Software Development @Internet Speed

Developing Quality Software @ Internet Speed: Toward "Next Generation" Models for Software Process Improvement

Why a Discovery Colloquium?

Competition in the new digital economy requires productive development of high-quality software systems at "Internet speed." However, relatively few Internet companies have been successful in developing software that combines quality and speed of production. Questions abound: What software development practices and models are effective in this new environment? What approaches may be most useful for rapid development? What is the right mix of quality, speed, and agility? What lessons from eXtreme Programming, Crystal, and other agile methodologies are useful? What are the critical questions about maintaining software, once developed, that impact business capability?

One highly placed CIO has credited software with some of the most spellbinding advances of the 20th century, but acknowledges that "software easily rates among the most poorly constructed, unreliable, and least maintainable technological artifacts invented by man." The Standish Group estimated bad software cost U.S. businesses $85 billion in lost productivity in one year alone.

To address these concerns, a one-day Colloquium on Software Development at Internet Speed will be held at the Software Engineering Institute on October 18, 2001. This Colloquium will explore issues faced by organizations that are challenged to develop quality software @ Internet speed. Participants will explore emergent trends and identify promising practices for next-generation software development.

The objectives are to:

- Expand understanding of the interface between business issues and Internet software development through open dialogue
- Discover and explore promising practices for Internet software development
- Analyze the range of views from participants to identify future directions for Internet software development

The Software Engineering Institute in partnership with Carnegie Mellon University and Georgia State University is sponsoring the Colloquium.

What Are the Driving Issues?

The explosion of electronic commerce onto the Internet is creating a radically different environment for software development. The core purpose of the Colloquium is to provide a forum for thoughtful dialogue on important issues related to software development.

Colloquium designers have identified an initial set of factors that stand out as compelling:

- **The separation between business and the software engineering system has closed:** In the digital economy, all parts of the company work hand in hand to achieve success, with emerging practices—e.g., convergent engineering and eXtreme Programming. Technology, strategy, and quality are increasingly integrated.
- **Market demands have become more complex:** Previously, software companies positioned themselves along a single dimension: either quality or time to market or cost. Now, competitive companies seek to compete with better products, delivered faster, and cheaper.
- **New approaches proliferate quickly and create tension between formal and informal practices:** Emerging practices are developing exponentially. As a result of Internet marketplace acceleration, the software engineering community has little formal knowledge on how to excel in high-speed development.
- **Software development risks are, more than ever, on the rise:** The constellation of risks to software development projects is larger and more complex. Errors and system failures amplify.

While these factors are significant, we expect that the Colloquium will identify additional factors: the Colloquium is designed to stimulate discovery.

Who Will Participate?

Approximately 35 invited participants have been selected based on their potential to contribute to the Colloquium discovery. Participants have been drawn from diverse areas—software practitioners from entrepreneurial and "brick and mortar" companies, researchers and Internet strategists, and software development experts. Colloquium participants represent the broad spectrum of stakeholders facing the critical challenges of Internet speed.

How Will Contributors Interact?

The Colloquium is intentionally designed to capitalize on the rich mix of participants and allow for maximum exchange of ideas (versus the traditional conference approach of speakers and workshops led by "experts"). The expectation is that all participants have expertise to share as well as openness to learn from one another.

The Colloquium will spotlight and encourage divergent views—not to drive wedges between people and companies but to build bridges of communication and knowledge, spurring unforeseen innovation and potential partnerships.

Using innovative open-forum search techniques, the Colloquium will stimulate rich exchange among participants. These techniques share several core principles:

- Involve a diverse group in shared inquiry and idea generation
- Encourage commitment to outcomes achieved rapidly
- Create an atmosphere of open dialogue, discovery, and respect for different views and knowledge
- Capitalize on this "discovery atmosphere" to forge mutually valued ideas and future directions

The Colloquium will apply these principles to advance inquiry, without reaching premature consensus in an emerging field. By employing search techniques, the Colloquium aims to galvanize ideas, identify promising practices, and create visions for the future.

Internet Speed and Quality as a "Moving Target"?

The idea of convening a Discovery Colloquium on Quality Software Development @ Internet Speed grew out of ongoing exploratory research on Internet-speed development. Conducted by a joint research team from SEI, Carnegie Mellon University, and Georgia State University, the study has evolved over the past year with a group of U.S. Internet companies, ranging from small startups to major brick and mortar Internet initiatives. Companies ranged in size from 20 to more than 300,000 employees. Industries included financial services, insurance, business and consulting services, courier services, travel, media, utilities, and government services.

Initial findings from the study revealed three important trends:

- Time drives development decisions.
- Quality depends on company practices, market pressures, and customer demand.
- Processes adjust as companies tweak their methods to achieve higher quality.

We anticipate the Discovery Colloquium will challenge and check initial findings, surface new insights, and significantly expand understanding of emerging problems and solutions for quality software development in the Internet-driven marketplace. In later phases of the study, the team plans to expand their research to include international companies for comparison with their U.S.-based findings.

Changing Market Conditions

Internet speed is a moving target subject to environmental factors and changing market conditions. Key factors include the nature of Internet companies themselves (e.g., dot.coms or brick and mortar) and emerging business models:

- B2B, B2C, C2C, and C2B strategies
- technology strategies
- industry-specific strategies

The "new economy"—"next economy" or "new new economy"—has taken another turn. When the Internet speed study began, the U.S. was experiencing the end-stages of the initial "gold rush" of the Internet. Now, Internet companies are retrenching and searching for business models that integrate competitiveness with reliability. The Colloquium organizers anticipate this moving landscape and encourage participants to bring their own observations and experiences.

What Will Participants Receive?

Attendees will have the opportunity to learn from one another and to help formulate new directions and innovative possibilities for quality software development in the fast-cycle Internet environment.

In addition to contributing to this vital exchange, participants will receive:

- Discovery Colloquium Report
- updates from the ongoing study team research on Internet speed
- access to a community for sharing problems, solutions, and potential partnerships
- invitation to participate in future events

Location

The Colloquium will be held at the Software Engineering Institute headquarters at 4500 Fifth Avenue, Pittsburgh, PA 15213.

Registration, Fees and Housing

The deadline for registration is August 31, 2001. There is a nominal charge to cover expenses of $150.00 per participant. Housing is not provided but recommendations for reasonable local housing are provided. See separate forms for information on registration and housing. Once registered, you will receive pre-read materials prior to the colloquium.

Appendix C Colloquium Agenda/Schedule

DISCOVERY COLLOQUIUM ON INTERNET SPEED

Software Engineering Institute
October 18, 2001

AGENDA

MORNING SESSION

Module I:	**Introductions and Opening of the Colloquium**
	Introduction to background and methodology for the Colloquium, Ground Rules, and introductions for all
Module II:	**Inclusion Exercise**
	Opening exercise in Breakout Groups
Break	
Module III:	**Panel Presentation/Discussion**
	Members of the study team discuss their findings, and set the stage for discovery learning, difference questioning, and contrasting points of view ("creative abrasion").
Module IV:	**Fishbowls: Listening Exercise in Sequenced Groups**
	Breakout Groups of participants in their contexts—types of organization and approaches to software development
	Series of fishbowls to allow dialogue in the spirit of open inquiry Full group listens as each fishbowl in turn disclose their thinking Each fishbowl discusses fully in their time allotment without cross-examination

Module V: **Discussion in Full Group: Surfacing Assumptions and Key Differences**

Colloquium placed on the "Timeline of Internet History"
Discussion in full group incorporates Study Team and fishbowl discussions to identify composite set of "Burning Questions"

Lunch

AFTERNOON SESSION

Module VI: **Exercise #1: Hypothesis Testing**

Breakout Groups identify observations, develop and test hypotheses on multiple / current worlds of Internet software development.

Module VII: **Exercise #2: Difference Questioning**

Breakout Groups surface assumptions underlying their hypotheses, and, without driving to premature consensus, extract "principles and practices" underlying their assumptions.

Break

Module VIII: **Exercise #3: Futuring/Scenario Development**

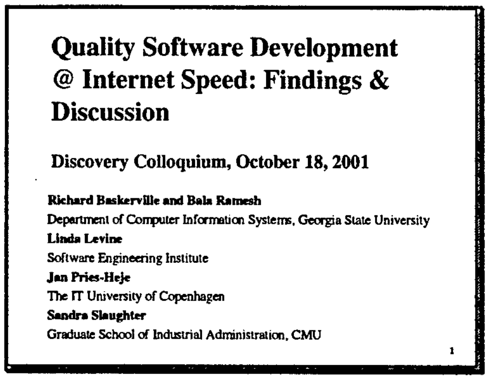
Facilitators will guide the full group to identification of emerging economic, political, technological, social, and other factors and their foreseeable impacts on changing the Internet environment. The full group will then select several probable scenarios of interest to the future of Internet software development. Breakout groups will create scenarios based on these factors and their impacts, together with contributions and constraints.

Module IX: **Final Session**

The Colloquium will conclude the day by identifying most promising and intriguing issues, and those fruitful for ongoing inquiry.

Appendix D Team Presentation Slides

Slides 1-2

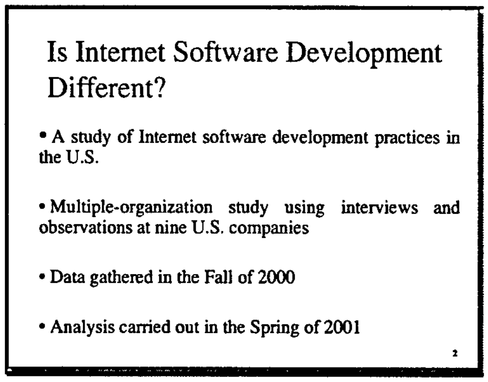


Quality Software Development @ Internet Speed: Findings & Discussion

Discovery Colloquium, October 18, 2001

Richard Baskerville and Bala Ramesh
Department of Computer Information Systems, Georgia State University
Linda Levine
Software Engineering Institute
Jan Pries-Heje
The IT University of Copenhagen
Sandra Slaughter
Graduate School of Industrial Administration, CMU

1

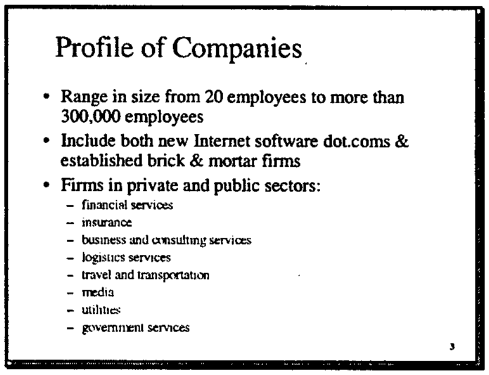


Is Internet Software Development Different?

- A study of Internet software development practices in the U.S.
- Multiple-organization study using interviews and observations at nine U.S. companies
- Data gathered in the Fall of 2000
- Analysis carried out in the Spring of 2001

2

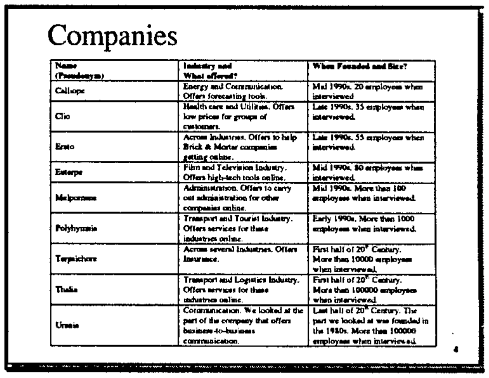
Slides 3-4



Profile of Companies

- Range in size from 20 employees to more than 300,000 employees
- Include both new Internet software dot.coms & established brick & mortar firms
- Firms in private and public sectors:
  - financial services
  - insurance
  - business and consulting services
  - logistics services
  - travel and transportation
  - media
  - utilities
  - government services

3

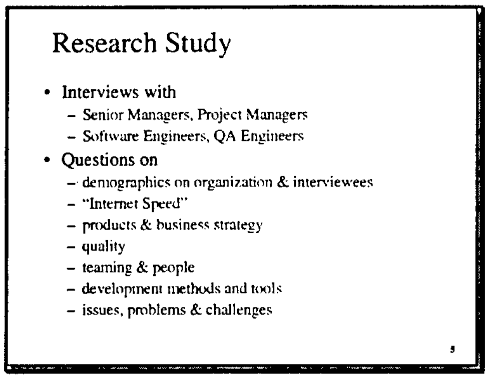


Companies

Name (Pseudonym)	Industry and What offered?	When Founded and Size?
Calliope	Energy and Communication. Offers forecasting tools.	Mid 1990s. 20 employees when interviewed
Clio	Health care and Utilities. Offers low prices for groups of customers.	Late 1990s. 35 employees when interviewed.
Erato	Across Industries. Offers to help Brick & Mortar companies getting online.	Late 1990s. 55 employees when interviewed.
Euterpe	Film and Television Industry. Offers high-tech tools online.	Mid 1990s. 80 employees when interviewed.
Melpomene	Administration. Offers to carry out administration for other companies online.	Mid 1990s. More than 100 employees when interviewed.
Polyhymnia	Transport and Tourist Industry. Offers services for these industries online.	Early 1990s. More than 1000 employees when interviewed.
Terpsichore	Across several Industries. Offers Insurance.	First half of 20th Century. More than 10000 employees when interviewed.
Thalia	Transport and Logistics Industry. Offers services for these industries online.	First half of 20th Century. More than 100000 employees when interviewed.
Urania	Communication. We looked at the part of the company that offers business-to-business communication.	Last half of 20th Century. The part we looked at was founded in the 1980s. More than 100000 employees when interviewed.

4

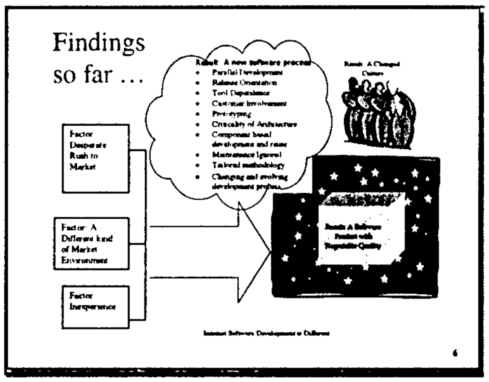
Slides 5-6



Research Study

- Interviews with
  - Senior Managers, Project Managers
  - Software Engineers, QA Engineers
- Questions on
  - demographics on organization & interviewees
  - "Internet Speed"
  - products & business strategy
  - quality
  - teaming & people
  - development methods and tools
  - issues, problems & challenges

5

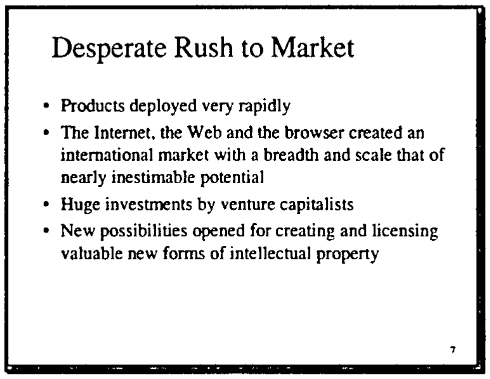


Findings so far …

Result: A new software process
Parallel Development
Release Orientation
Tool Dependence
Customer Involvement
Prototyping
Criticality of Architecture
Component based development and reuse
Maintenance Ignored
Tailored methodology
Changing and evolving development process
Result: A Changed Culture
Factor: Desperate Rush to Market
Factor: A Different kind of Market Environment
Factor: Inexperience
Result: A Software Product with Negotiable Quality
Internet Software Development is Different

6

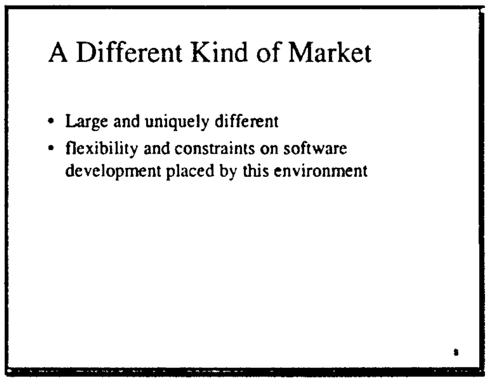
Slides 7-8



Desperate Rush to Market

- Products deployed very rapidly
- The Internet, the Web and the browser created an international market with a breadth and scale that of nearly inestimable potential
- Huge investments by venture capitalists
- New possibilities opened for creating and licensing valuable new forms of intellectual property

7

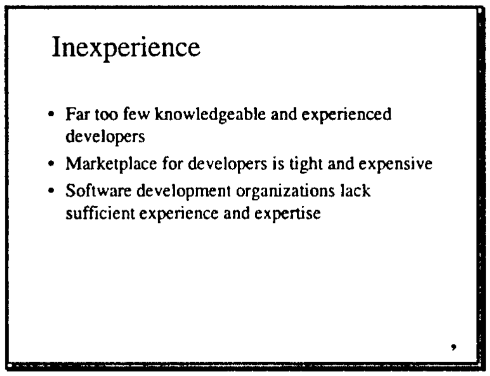


A Different Kind of Market

- Large and uniquely different
- flexibility and constraints on software development placed by this environment

8

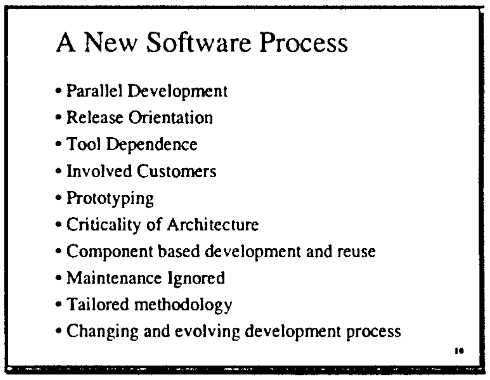
Slides 9-10



Inexperience

- Far too few knowledgeable and experienced developers
- Marketplace for developers is tight and expensive
- Software development organizations lack sufficient experience and expertise

9

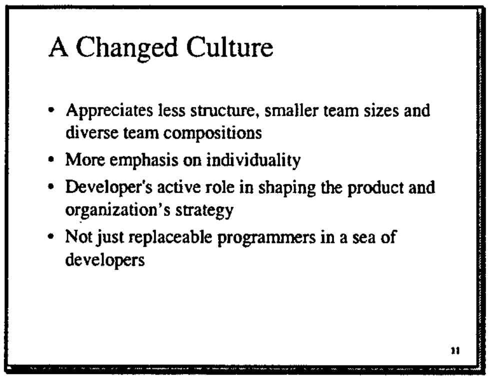


A New Software Process

- Parallel Development
- Release Orientation
- Tool Dependence
- Involved Customers
- Prototyping
- Criticality of Architecture
- Component based development and reuse
- Maintenance Ignored
- Tailored methodology
- Changing and evolving development process

10

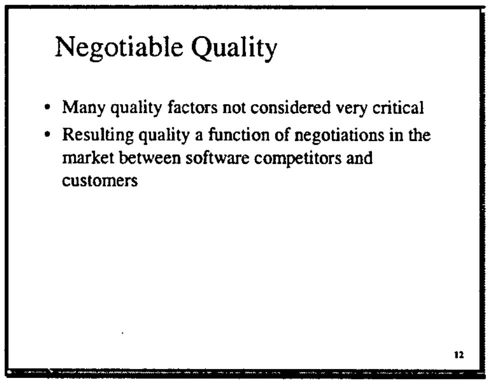
Slides
11-12



A Changed Culture

- Appreciates less structure, smaller team sizes and diverse team compositions
- More emphasis on individuality
- Developer's active role in shaping the product and organization's strategy
- Not just replaceable programmers in a sea of developers

11

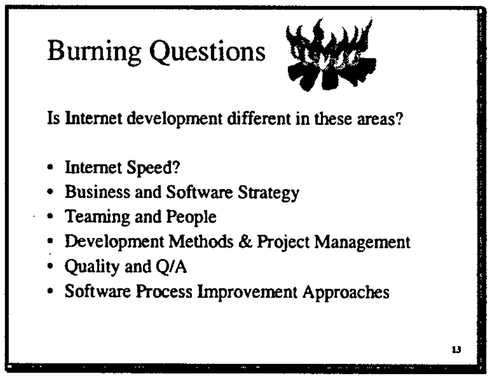


Negotiable Quality

- Many quality factors not considered very critical
- Resulting quality a function of negotiations in the market between software competitors and customers

12

Slides
13-14

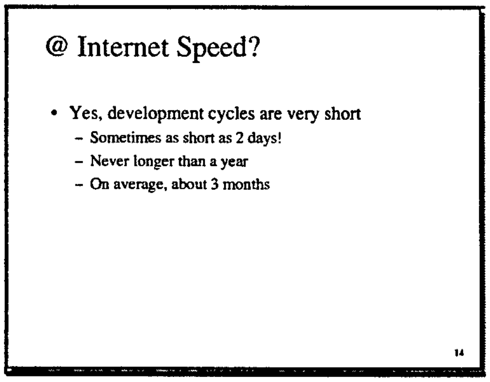


Burning Questions

Is Internet development different in these areas?

- Internet Speed?
- Business and Software Strategy
- Teaming and People
- Development Methods & Project Management
- Quality and Q/A
- Software Process Improvement Approaches

13

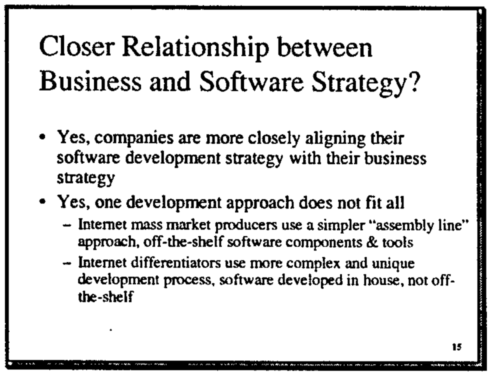


@ Internet Speed?

- Yes, development cycles are very short
  - Sometimes as short as 2 days!
  - Never longer than a year
  - On average, about 3 months

14

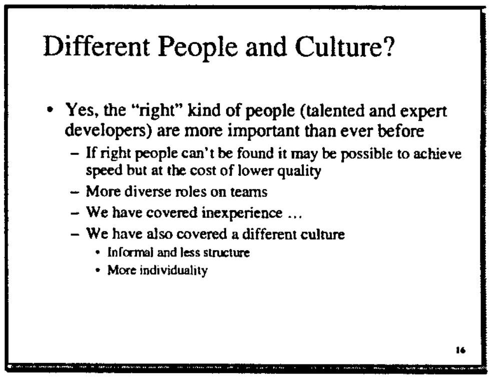
Slides
15-16



Closer Relationship between Business and Software Strategy?

- Yes, companies are more closely aligning their software development strategy with their business strategy
- Yes, one development approach does not fit all
  - Internet mass market producers use a simpler "assembly line" approach, off-the-shelf software components & tools
  - Internet differentiators use more complex and unique development process, software developed in house, not off-the-shelf

15



Different People and Culture?

- Yes, the "right" kind of people (talented and expert developers) are more important than ever before
  - If right people can't be found it may be possible to achieve speed but at the cost of lower quality
  - More diverse roles on teams
  - We have covered inexperience ...
  - We have also covered a different culture
    - Informal and less structure
    - More individuality

16

Appendix E Group Worksheets

DISCOVERY COLLOQUIUM ON INTERNET SPEED
Software Engineering Institute
October 18, 2001

WORKSHEETS and "The Pig's Tale" TEMPLATE

Definitions

Observation: **Description of an "as-is" state or result that exists in the Internet software community.**

Hypothesis: Broad suppositions (e.g., about cause and effect, correlations, and solutions) regarding the observation.

Assumption: Unexamined reasons about why the hypotheses are true; these may include a wide range of drivers/causes, effects/consequences, catalysts, conditions, contexts, world views, beliefs, values, etc.

WORKSHEET #1: HYPOTHESIS TESTING

Step 1: Observation on Software Development at Internet Speed

a. As a group, from all the observations in the morning (study team findings, fishbowls, inferred from a comment or question, etc.), choose an observation related to Internet speed development that you find most compelling. b. Write your observation(s) on your flip chart.	**Observation:**

Example: **Three little pigs successfully thwart the wolf.**

Step 2: Hypothesis Development

a. Together, brainstorm a set of possible hypotheses that are inextricably tied to and explain why this state or condition exists. b. Once you have decided on your complete list of hypotheses, jot the brainstormed items up on the flip chart, leaving space for further comments under each one. c. If you have more than one flip chart of hypotheses, post them on the wall side by side.	Observation: **Hypothesis:** 1. 2. 3. 4.

Example:
Hypothesis 1: Pig 3 had performed excellent risk analysis.
Hypothesis 2: Pig 3 built and tested house 3.0.
Hypothesis 3: All three pigs used combined weight and strength to support door.
Hypothesis 4: Wolf was tired of blowing down houses.
Hypothesis 5: House version 3.0 has better feng shui.

Step 3: Hypothesis Testing

a. Once the hypothesis set is developed, decide as a group whether you want to keep all of them. Some may be those "wild" ideas created in the spirit of brainstorming and not of interest to your group for further examination. Beware of throwing out the "wild" ideas too quickly, however. They could lead to issues of interest. There is no need to narrow the list unless you want to; the focus here is on generating all the ideas that you want to consider. b. Are there hypotheses that appear to be mutually contradictory ("Devil's Advocate" hypotheses)?	**Analysis:** (Notes on your rationale, what hypotheses are rejected and which not, and why, etc.)

Example:

a. Group decides to retain all 5 hypotheses for examination.

b. Hypotheses 1 and 2 seem aligned; 3 seems to contradict the "self-made pig" of 1 & 2 and emphasize teamwork; 4 looks at the market/competition; and 5 looks at intrinsic qualities in the house itself, and has commonalities with 2.

WORKSHEET #2: DIFFERENCE QUESTIONING

Step 1: Assumption Identification

a. Together as a group, for each hypothesis, identify the underlying assumptions (or "foundational choices") or differences among points of view as represented by the hypotheses. b. Write your assumptions under each hypothesis on your flip chart. c. There will probably be several assumptions under each hypothesis. d. There will probably be several assumptions under each hypothesis.	Hypothesis: Assumptions: 1. 2. 3. Hypothesis: Assumptions: 1. 2. 3.

Example:
Hypothesis 1: Pig 3 had performed excellent risk analysis
  Assumption 1: Risk analysis can make a difference
  Assumption 2: Pig 3 is smarter than Pig 1 or Pig 2
  Assumption 3: Pig 3 had time to perform risk analysis before wolf eating
Hypothesis 2: Pig built and tested house 3.0
  Assumption 1: First 2 houses blew down
  Assumption 2: Pigs were able to learn from failures v1.0 and v2.0
  Assumption 3: Straw is less effective than wood
  Assumption 4: Wood is less effective than brick
Hypothesis 3: All 3 pigs used combined weight and strength to support door
  Assumption 1: 3 or more pigs are required to defeat the wolf
  Assumption 2: Teamwork prevails
Hypothesis 4: House version 3.0 has better feng shui
  Assumption 1: Energy of the environment is a factor in house strength
  Assumption 2: The house can stand alone on its own without pig intervention

Step 2: Draft - Principles and Practices

a. Look for patterns of commonalities or differences across the assumptions. Note those that recur or contradict. Look for principles or practices that derive from your assumptions.	**Principle # 1**
b. Create a list of the principles and practices that your group sees as significant.	**Principle # 2**
c. You need not come to consensus on a limited list of these principles.	**Principle #3**
d. As your group identifies principles, questions will also emerge. Note these on your flip chart as well.	**Emergent Questions:**

Example:

Principle #1: A variety of approaches can be effective in thwarting the wolf

Principle #2: Heroes, experience, and teamwork all seem able to yield the desired result

Principle #3: Unexpected factors can have strong effects—e.g., wolf tiring or feng shui

Principle #4: There may be decision criteria larger than simply achieving the result that are important factors for organizational (market) strength.

Open/Emerging Questions:

What are the relative costs and benefits of heroes, experience, teamwork?

What are the decision criteria and how are they arrived at? What underlying factors are important?

What kinds of organizational agility are desirable to meet with unexpected events successfully?

Identify any "promising practices" (tried or untried) that emerged from the discussion. Come back to the corral to add other practices throughout the day.

Identify both the results and the group process and thinking your group used to reach your findings.

Examples of Promising Practices:

- **Pigs are more effective at dealing with wolves if they have structural analysis of their homes.**
- **Pigs work better when they share resources rather than working alone.**

Promising Practices Corral:

Step 3: Create a Mini-Report from Your Group

WORKSHEET #3: FUTURING/SCENARIO DEVELOPMENT

Step 1: Futuring: The New Internet World

a. Together as a full group, all participants identify: What are the emerging economic, political, technological, social, and other conditions in the world that will affect the Internet and software development over the next 2–3 years?
b. What will these changing conditions impact most significantly?

Example:

Emerging Conditions	**Impacts**
Hungry wolves increase in number	**Pigs need to communicate across long distances**
Housing industry experiences market renewal	**Online sales of pre-fabs explode**
Pigs realize their need to share experience	**Reliable knowledge management skills in increasing demand**

Emerging Conditions	**Impacts**

c. From these conditions and impacts, participants in full group will identify a set of probable scenarios for the Internet and software development that are likely to emerge in the next two to three years.
d. From the set of scenarios, facilitators will guide the full group to select four possible scenarios to examine in their breakout groups, two for each group.

Step 2: **Scenario Development: Forces Affecting What Is Emerging**

a. Each of the two breakout groups works with two probable scenarios.

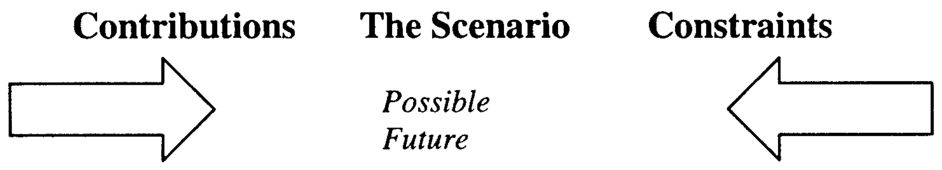
Example: Probable scenarios:

1) **The expanded demand for reliable housing and information amplifies the need for software that provides "full-service housing"—from how-to instructions to sales of pre-fabs**
2) **The preponderance of unexpected and unpredictable situations leads to increased focus on intelligent agents research to explore this field**

b. Each scenario is articulated through Force Field Analysis, with the probable scenario as the pivot.

c. For each scenario, the group will identify **contributions**—fostering practices and forces—and **constraints**—hindering practices and forces.

Example: Force Field



Pig knowledge about wolf danger	**The Scenario: Expanded demand for reliable housing & info amplifies software to provide "full-service housing"**	**Pig skepticism about wolf danger**
Software manufacturers looking for new, security-based products		**Security software has higher reliability/quality requirements than housing software developers generally expect**
Pigs who lived collaborate to form new company "Huff & Puff"		**Wolves promote "disinformation" campaign**
Government subsidies available for pig housing		**Pigs unaware that interest in their safe housing is by humans are also interested in eating them in due time**

An alternate approach to generating this data could be Mind-Mapping, should participants so wish to approach this exercise.

Force Field Analysis Matrix

Contributions	The Scenario	Constraints

Contributions	The Scenario	Constraints

Step 3: Create a Report from Your Group

a. The entire Force Field Analysis (or Mind-Mapping) will be captured on each group's flip chart.
b. Identify both the results and the group process your group used to reach your findings.
c. Have someone in your group ready to report out to the full group.

FINAL SESSION AND DISCOVERY OF ONGOING FRUITFUL INQUIRY

To close the day, the full group will discuss and identify the following:

1. **What were the most promising ideas/critical barriers?**

2. **What were the most intriguing/important questions?**

3. **What are the areas where participants would like to see as most fruitful for ongoing inquiry?**

4. **What's really new? Anticipated? Uncertain?**

References

[Collins 97]	Collins, J. & Porras, J. *Built to Last: Successful Habits of Visionary Companies.* New York, NY: Harperbusiness, 1997.
[Goldstein 94]	Goldstein, J. *The Unshackled Organization: Facing the Challenge of Unpredictability through Spontaneous Reorganization.* Portland, Oregon: Productivity Press, 1994.
[Leonard 99]	Leonard, D. *When Sparks Fly: Igniting Creativity in Groups.* Cambridge, MA: Harvard Business School Press, 1999.
[Weisbord 95]	Weisbord, M. *Future Search: An Action Guide to Finding Common Ground in Organizations and Communities.* San Francisco: Berrett-Koehler Publishers, 1995.

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.		
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE September 2002	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Discovery Colloquium: Quality Software Development @ Internet Speed		5. FUNDING NUMBERS F19628-00-C-0003
6. AUTHOR(S) Linda Levine, Richard Baskerville, Jo Lee Loveland Link, Jan Pries-Heje, Balasubramaniam Ramesh, Sandra Slaughter		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2002-TR-020
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2002-020
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

In October, 2001, the Software Engineering Institute hosted a colloquium to explore issues faced by organizations challenged with developing quality software at "Internet speed" and to examine the impact of Internet speed on current software-development practices. The colloquium was an element of an exploratory study being conducted by a team of researchers from Carnegie Mellon University, Georgia State University, and The IT University of Copenhagen. Team members shared their findings from a previous phase of the study in which they had investigated quality and fast-cycle Internet-software engineering in nine organizations. Members of those organizations attended the colloquium, along with others who were invited to participate based on their perceived ability to contribute to the discussion of Internet-speed development.

This report describes the activities of the colloquium and the raw data collected during group discussions and breakout sessions. It also includes a brief description of the field study done by the cross-institutional team. A preliminary analysis of the results of the colloquium is presented in the conclusion of the report.

14. SUBJECT TERMS Internet speed, Internet software development, software quality			15. NUMBER OF PAGES 98
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL